LEVEL III

Bolt Beranek and Newman Inc.

AD A109648

12

A092971

Report No. 4785

Research in Knowledge Representation for Natural Language Understanding

Annual Report

1 September 1980—31 August 1981

DTIC
ELECTE
JAN 15 1982
S
D
E

Prepared for:
Advanced Research Projects Agency

DTIC FILE COPY

8

This document has been approved for public release and sale; its distribution is unlimited.

82 01 13 077

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 4785	2. GOVT ACCESSION NO.: AD-A109 648
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING Annual Report	5. TYPE OF REPORT & PERIOD COVERED: Annual Report 9/1/80 - 8/31/81
	6. PERFORMING ORG. REPORT NUMBER: 4785
7. AUTHOR(s): C.L. Sidner, M. Bates, R.J. Bobrow, R.J. Brachman, P.R. Cohen, D.J. Israel B.L. Webber, and W.A. Woods	8. CONTRACT OR GRANT NUMBER(s): N00014-77-C-0378
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research Department of the Navy Arlington, VA 22217	12. REPORT DATE: November 1981
	13. NUMBER OF PAGES: 244
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial intelligence, knowledge representation, natural language understanding, discourse, parallel algorithms, planning, reference, intended meaning, parsing, semantics, lexicon, cascade, marker passing, syntactic-semantic interactions, cognitive science, KL-ONE, RUS, PSI-KLONE, structural inheritance networks.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes the research of BBN's ARPA-sponsored Knowledge Representation for Natural Language Understanding project during its fourth year. In it we report on advances, both in theory and implementation, in the areas of knowledge representation, natural language understanding, and abstract parallel machines. In particular, we report on theoretical advances in the knowledge representation system KL-ONE, extensions to the KL-ONE system, and new uses of KL-ONE in the domain of

cont'd.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060100

Abstract (cont'd.)

knowledge about graphic displays. We report on a design for a new prototype natural language understanding system, on issues in cascaded architectures for interaction among the components of a language system, and on a module for Lexical acquisition. In addition, we examine three topics in discourse: a new model of speaker meaning, which extends our previous work on speakers' intentions, an investigation of reference planning and identification, and a theory of "one"-anaphora interpretation. Our discussion of abstract parallel machines reports on a class of algorithms that approximate Quillian's [49] ideas on the function of human memory.

Report No. 4785

RESEARCH IN KNOWLEDGE REPRESENTATION FOR NATURAL LANGUAGE UNDERSTANDING

Annual Report
1 September 1980 to 31 August 1981

Principal Investigator:
William A. Woods
(617) 497-3361

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification DD form 1473
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

ARPA Order No. 3414

Contract No. N00014-77-C-0378

Effective Date of Contract:
1 September 1977

Contract Expiration Date:
1 February 1982

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-77-C-0378. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Bolt Beranek and Newman Inc.'s (BBN's) ARPA project in Knowledge Representation for Natural Language Understanding is developing techniques for computer assistance to a decision maker who is collecting information about and making choices in a complex system or situation. In particular, we are designing a system with natural language control of an intelligent graphics display of the kind that can be used in a command and control context, both in strategic situation assessment and in more tactical situations. We believe that the commander needs an extremely flexible system, capable of manipulating large amounts of data and presenting them on a graphical display in a variety of ways until the commander feels satisfied that he has a grasp of the situation. Such a system would be able to

- display many kinds of different map overlays
- change the kinds and amounts of detail shown
- construct unique kinds of displays to suit the situation at hand
- display tabular and graphical information and present textual material in ways that are easily comprehensible.

Techniques to produce such displays on demand, in response to high-level specifications of what they should contain, require significant breakthroughs in areas of language understanding, knowledge representation, and knowledge-based inference. Our work falls into three classes, motivated by the goal of providing powerful computer assistance to a commander in a complex

decision-making task. These areas are: fluent natural language understanding in a graphics context (including intelligent, helpful systems that go beyond mere passive execution of literal instructions), fundamental problems of knowledge representation and use, and abstract parallel algorithms for knowledge-based inferential operations.

The major accomplishment of our work so far is the development of the knowledge representation system KL-ONE. Another accomplishment is a prototype display system that understands requests, and some assertions, about display manipulation. The language understanding prototype is based on several general tools, including the RUS parser, a lexical acquisition system, the PSI-KL-ONE interface, and a model of a speaker-meaning recognizer. In addition, we have pursued research on an abstract parallel machine for marker passing and on implementations of knowledge-based inference algorithms.

In the natural language understanding (NLU) system, the KL-ONE knowledge representation system has been used to represent the results of syntactic analysis, to organize the semantic interpretation rules used to interpret sentences (in the PSI-Klone interface), to organize models of the user's plans and goals (in the speaker meaning recognizer), to specify the information in the domain knowledge base, and to codify the rules needed for drawing pictures on the graphic display. The knowledge-structuring capabilities of the KL-ONE system have

proven themselves very powerful in this system, and the extent to which the same structures have proven useful in qualitatively different parts of the system gives evidence of the robustness of these capabilities.

The tools we have been researching and building have been used not only in our own work but in the ARPA-sponsored CONSUL group at Information Sciences Institute (ISI). We have provided them with versions of the RUS parsing system and the KL-ONE knowledge representation system. We expect that the lexical acquisition system will soon be available for their use as well. Knowledge representation problems arising out of that work have been important in our new work on KL-ONE, and the interaction of the two groups has benefited both. We continue also to cooperate with the ARPA-supported AIPs project [28] at BBN, which is using KL-ONE. Other uses of KL-ONE are discussed in an upcoming subsection on the KL-ONE community. Versions of the RUS parser are also currently in use at the General Motors Research Lab and by the medical decision-making group at the National Library of Medicine.

In this report, we describe the research investigations we have accomplished in the three main areas we are pursuing - parallel algorithms, representation research on KL-ONE, and natural language research. Chapter 2 first updates our 1979 report [13] by discussing extensions and changes in the system since that time. In Chapter 3 we present an overview of the

Natural Language system, followed by two chapters on specific parts of that system. One chapter, on the PSI-Klone interface, contains a discussion of changes and extensions to RUS and PSI-KL-ONE. It also presents our new work on a lexical acquisition system. The other chapter contains a discussion of our work on issues related to discourse, primarily research on models of speaker meaning, on reference as a planned act and on one-anaphora. The final chapter presents research on the development of algorithms for abstract parallel machines. To conclude, we give a summary of the research papers we have written and presented on our work.

2. A REPORT ON KL-ONE

2.1 KL-ONE Overview and Philosophy

by R. J. Brachman and D. J. Israel

KL-ONE is the underlying representational formalism for the Natural Language Understanding System being designed and implemented at BBN. It is a semantic net formalism based on the ideas of structured inheritance networks and, correlatively, of complex, structured concepts. In this brief overview, we shall confine ourselves chiefly to the second of the above two leading themes in the philosophy behind KL-ONE. (For a more general introduction, see Brachman's "An Introduction to KL-ONE" in [13]. See also [8], [9], [10]). In the following sections, we shall only briefly note the most important ways in which the current version of KL-ONE differs from the system described in that earlier annual report. KL-ONE was originally designed by R. Brachman; R. Brachman and D. Israel completed the work discussed here, with contributions from R. Bobrow, J. Schmolze, and B. Woods.

There is now a large number of semantic-net formalisms; the designers of most of these have been heavily influenced by Minsky's conception of "frames" [38]. Almost uniquely, the semantics of KL-ONE is not frame-based. (But see also [60], [57], [18], [58].) Indeed, it is crucial to a correct understanding of KL-ONE that the differences between it and

standard frame-based networks be kept in mind. More particularly, it is crucial that the reader understand our reasons for being dissatisfied with such frame-based systems.

2.1.1 Background: Knowledge Representation

Before contrasting KL-ONE with other semantic net formalisms, it is perhaps advisable to set this comparison within the broader context of current work in knowledge representation, especially given the enormous and enormously confusing variety of work being done under that rubric in AI. Alan Newell, in his 1980 AAAI Presidential Address, commenting on the SIGART Special Issue in Knowledge Representation ([15]), notes that

> The main result was overwhelming diversity - a veritable jungle of opinions. There is no consensus on any question of substance. Brachman and Smith themselves highlight this throughout the issue, for it came as a major surprise to them. Many (but of course not all!) respondents felt the same way... What is so overwhelming about the diversity is that it defies characterization... But there is no tidy space of underlying issues in which respondents, and hence the field, can be plotted to reveal a pattern of concerns or issues.

[40] We shall here attempt to provide a brief guide through some of this jungle of distinguishing the very different kinds of activities within the field of knowledge representation.

The first distinction that must be observed is between representational formalisms or languages and tools for building such formalisms. At the most abstract level, work done on various aspects of various dialects of LISP - for example, efforts to

accommodate abstract data types more naturally - can be seen as aimed toward the design of tools for representational-formalism-implementors; but though many (indeed most) such formalisms are in fact embedded in LISP, very few (and perhaps no) researchers in the field think of LISP as a representation language. On a more concrete level are tools for building representational systems that are meant to be independent of the choice of programming language. We would classify RLL [29] and also, though more tentatively, MRS [27] as belonging in this category. There is an analogy between this kind of research and the work done on developing tools for building expert systems, tools which need not themselves take the form of expert systems.

When we move over to representational languages proper, there is a major dichotomy between those designed, first and foremost, to accord with some explicitly articulated semantic theory meant to provide a standard of correctness for the implemented interpreter of the formalism and, on the other hand, those which eschew such theories (again more or less explicitly). [Needless to say, this dichotomy is somewhat controversial.] Among the languages that eschew semantic theories, there is another important dichotomy - between those whose design is motivated and guided by some account of human psychology, in particular of human cognitive processing, and those for which there is no theory independent of the design, to which the system is supposed to be faithful. The original work in semantic nets of Quillian ([48], [49]) and Collins and Quillian ([51]) was

explicitly driven by the desire to produce a model for certain observed psychological regularities - in particular regularities of access and retrieval from long-term memory. (See [34].) One could argue that the most ambitious project in this vein was KRL [7]. On the other hand, systems such as FRL [53] and [64] were not designed to "simulate" natural cognitive processes; rather once their access, retrieval, and data structure-manipulating operations are specified, there is no further question of whether those are the right operations. Such systems can be judged as good or bad along various dimensions; but there is no independent account of any domain, other than the explicitly computational, that they are meant to capture.

Finally, among those representational systems meant to embody in a computationally tractable way some theory of meaning (some semantic account for the constituent language), there is yet again a distinction. First are those whose language is specified to be of a standard quantificational sort (usually first-order) in its standard syntactic clothing and, most important, with its standard Tarski-style model-theoretic semantics. (The role of the model-theoretic semantics is central here. Typically, the system's interpreter is a theorem-prover that realizes a sound, and perhaps complete, proof-procedure for the language <u>according to that particular semantic account</u>. For more on this topic, see [34].) On the other hand, there are now a number of representational formalisms whose constituent data structures are graph-theoretic and whose basic operations are

graph-oriented. The semantic theories of such formalisms need not be identical with the classical model-theoretic account of quantificational formalisms, especially first-order formalisms. Within this group, we would place MANTIQ [63], NETL [24] (though this is a mixed case owing to the importance of a certain range of psychological considerations in the background of its design), Shapiro's SNePS [60], the systems of Schubert [57] and Schubert, Cercone and Goebel [58], and KL-ONE.

2.1.2 Frames

Let's assume that we are talking about a typical representation scheme based on the notions of frame-like conceptual units and slot-like role descriptions, or their equivalents, and a hierarchy for passing properties from general to specific conceptual units. The principal connective (typically "ISA") makes one conceptual unit a subcategorization of another. The "inheritance of properties" (or the notion of "virtual copy") in such a scheme means that subconcepts inherit all of the features of their parent concepts.

Consider the simple frame/concept ELEPHANT, which we might portray as in Figure 1. The knowledge representation literature tells us that elephants are mammals, that they are gray, and that they have four legs. So we would see an "ISA" connection between the concept ELEPHANT (the ellipse labeled as such in the figure) and the concept MAMMAL, with a role description (or with four descriptions, perhaps) on ELEPHANT signifying that the number of

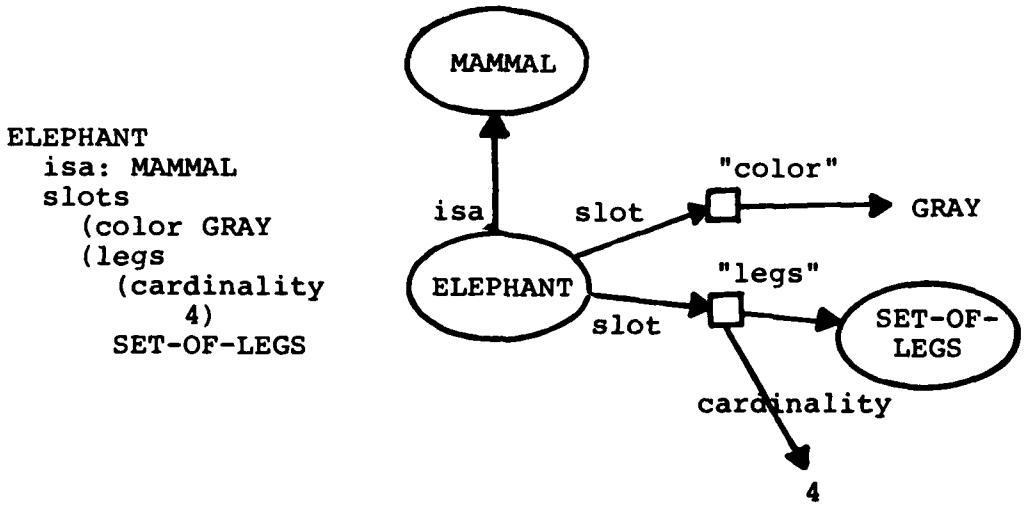


FIG. 1. A SIMPLE FRAME FOR "ELEPHANT"

legs on an elephant is 4, and a filled-in role description saying that the color of an elephant is gray (the role descriptions, or slots, are pictured as small squares). One might be tempted to an informal reading of this as "an ELEPHANT is a MAMMAL that has four legs and whose color is gray," and, implicitly, "every ELEPHANT is a MAMMAL that has ..."

2.1.3 Is nothing sacred?

The principal inference mechanism in these representation systems is "inheritance of properties". It usually runs something like this: if Clyde is asserted to be an elephant by some version of an "ISA" connection from CLYDE to ELEPHANT, then the node CLYDE would act as if it were a copy of the entire

structure at the ELEPHANT node. This being the case, we could conclude that Clyde is a mammal, the value of his color attribute is gray, and he has four legs.

Since these attributes accrue to Clyde in virtue of his being an elephant, slots (as well as inherited ISA links) seem to correspond to the consequents of if-then statements. If you fit the prototype, then you get the properties. One doesn't check the properties first to see if there is an instance of a frame; rather, properties expressed by slots follow from instantiation of the frame. This interpretation is substantiated in various places in the literature. Charniak [19] proposes a combination of frames and predicate calculus notation in which he translates a frame into a universal statement, with "implicit existentials" for the slots. For example, Charniak might translate the frame

```
[elephant
      isa:  (mammal ?elephant)
      slots: (head (elephant-head ?head))
      facts: (has-part ?elephant ?head) ...]

     into
FORALL (?e (elephant ?e))
   EXISTS (?h (elephant-head ?h))
          [has-part ?e ?h] ...
```

In a similar vein, Hayes ([33]) interprets the slots of frames as the right-hand sides of conditionals. In his view, the frame represents implicitly universally quantified conditionals. For example, "a frame representing the concept C, with slot-relationships $R_1,...,R_N$ " becomes the following (expressed in clausal form, without Hayes' criteriality assumption):

$$\forall x\ C(x) \Rightarrow R_1(x, f_1(x))$$
$$\& \forall x\ C(x) \Rightarrow R_2(x, f_2(x))$$
$$\& \ .\ .\ .$$

But not all instances of a concept like ELEPHANT need have the features specified there. For example, it is quite conceivable that we might run across some poor elephant that had only three legs (perhaps an unfortunate veteran of the Punic Wars). To handle exceptions like three-legged elephants, some notations allow the "cancellation" of properties that a particular concept would normally inherit. So the intuitive reading of the ELEPHANT frame as including "every elephant has four legs" is not quite right. Rather, it should be that typically (in the sense of a default) elephants have those properties. Thus, as Fahlman puts it, "I am using a weak sense of the word 'every' here: I mean that the property is true of every elephant for which it is not explicitly cancelled" ([24] p. 16). Indeed, Fahlman has gone so far as to call his node "TYPICAL-ELEPHANT".

Now, this interpretation would not be too bad if one were to believe that a frame "is intended to represent a 'stereotypical situation'" ([24], p. 48). We just have to keep in mind that any notation that allows us to represent exceptions by cancelling properties that would normally be inherited must be using its

conceptual units in this sense. (The concept of an X is really the concept of a typical X.)

Notice that the implied sense of "typically" here is strictly along the lines of "'in the absence of any information to the contrary, assume ...'" ([24], p. 210), and has nothing to do with frequency of occurrence. In some dialects, "typically" is more closely synonymous with "usually". In the manner used in frame notations, however, a "typical" property could be violated in every single case! We do not treat the "usually" issue further, but merely note that it is yet another type of adverb of quantification that we might like to express.

2.1.4 Universal statements

One consequence of the default interpretation of concepts is that the slot notation for properties cannot be used, without alteration, to make an unequivocal universal statement (since we hedged on the "every"). And we certainly want to be able to make exceptionless universal statements, since there are many domains and situations where properties do hold for all instances of certain kinds.

For example, all banks in Massachusetts happen to be closed on Sunday. It would be most convenient to represent this contingent universal fact with a single statement. One might suppose that we could use the default notation as it stands, and examine the set of instances to see if a concept represented a

true universal. But this procedure would only be useful in knowledge bases where the information was expected to be complete. (See [52] for more on the "closed world assumption", which presumes "perfect knowledge about the domain being modeled"). And even then, checking all instances could be very costly.

At least one "proto-net" author has suggested a way to represent exceptionless properties explicitly: "...some fact [can be declared] sacred (uncancellable) and therefore true of all elephants without exception" ([24], p. 16). But without such an augmentation, no notation that allows cancellation of properties can express universal truths.

Even if we admit "sacred" properties as a way to express universals, we are still missing a distinction. Notice that Massachusetts banks being closed on Sundays is a matter of (contingent) fact; it just happens to be the case at the current time. In this instance, we implicitly understand the possibility of things being otherwise. There is an important distinction between statements of contingent universal fact and necessary truths. A "sacred" statement, which says there are no exceptions, is different from one that says there can't be any exceptions.

Given the previous observation that slots express material conditionals, it seems reasonable that necessary properties might be expressed by representing the necessity of those conditionals

- i.e., with valid statements rather than just true ones, though as far as we know, this is not generally done. In any event,one can't have one's cake and eat it, too: one's slot notation per se must mean one thing or the other.

2.1.5 Sufficiency and composite concepts

We must also distinguish those necessary truths whose necessity rests on definitional equivalence from those whose necessity is, in a sense to be explained, primitive. Contrast the concepts QUADRILATERAL and ELEPHANT. Something is a quadrilateral if and only if it is a polygon and has four sides. But even if elephants necessarily were mammals that had four legs, we could not conclude that any four-legged mammal was necessarily an elephant. That is, while the property, "four-legged mammal", may be necessary, it is certainly not sufficient for being an elephant. In fact, it is probably impossible to come up with the complete story of what it means to be an elephant - in other words, "natural kind" concepts cannot be criterially defined. In contrast, there is nothing more to the story of quadrilaterals than four-sidedness on top of "polygonicity". (See [47].)

So we must face up to another distinction: some concepts are associated with natural kinds and some are not. Or, at the very least, we must be able to specify when certain properties are criterial (i.e., when sets of them are analytically sufficient to determine instantiation of a given concept).

Many researchers have expressed doubts as to the importance of definition ; for instance, many have said that no lexical items of a natural language have analytic definitions, not even "bachelor" (e.g., see [25]). Hence, the distinctions between contingent and necessary truths and between definitionally necessary truths and non-definitional necessities may seem like logical nit-picking. After all, even if one admits that bachelors and quadrilaterals are definable, one might further argue that the vast majority of interesting terms are like "elephant", for which criterial definition is impossible (e.g., "... aside from mathematics and the physical sciences, most of what we know about the world has associated exceptions and caveats" [52] p. 218). But consider this: once you have the concept of an elephant - natural kind or not - you can define the concept of an elephant with three legs without any trouble - as simply an elephant the number of whose legs is exactly three. No mystery there. In fact, this is a composite concept directly analogous to QUADRILATERAL (a.k.a. POLYGON-WITH-FOUR-SIDES), which is simply the concept of a polygon the number of whose sides is four. We are equating the atomic term "quadrilateral" to that composite concept. Of course, the basic natural kind does not work like this - "elephant" does not mean "mammal with four legs", and we are not merely attaching the name "elephant" to a concept like MAMMAL-WITH-FOUR-LEGS. (That an elephant is a mammal is true by virtue of fact; that a quadrilateral is a polygon is true by virtue of necessity.) ELEPHANT-WITH-THREE-LEGS, however, is like QUADRILATERAL, and not like ELEPHANT.

The lesson here is that in order for a knowledge representation system to be able to handle all kinds of concepts - even the simplest composites constructed from natural kind-like concepts - some form of definitional structuring is necessary. The internal structure of non atomic concepts (e.g., proximate genus and essential difference) must be transparent to the interpreter and definitional, or else the interpreter can't tell if the network creator is lying about that composition. Note that this does not imply that any lexical items need have analytic definitions, only that one should not be able to cast any doubt on a three-legged elephant's being an elephant.

Unfortunately, the way most network representations are constructed, defaults are adopted at the expense of definition. Slots/roles are not parts of the "meaning" of a frame - they are only properties that (typically) follow from being an instance of the frame. The direction of the role descriptions is always outward: something has to be described by a concept before one can see what properties follow from that attribution. As a result, the interpreter cannot use the structure of concepts to determine subsumption of one concept by another. That is, the system cannot tell if something is a specialization of another concept, even if that fact should be transparent from its content. For example, rather than describe the composite concept of a rhombus as a polygon with four equal-length sides - which should make self-evident its relation (by virtue of meaning) to the concept of a quadrilateral - we are forced to create the

empty frame RHOMBUS, and later, tell the system that this frame "ISA" QUADRILATERAL as a separate fact. Thus, in effect, every concept in the network is primitive. Concepts are as good as atomic, and you must explicitly tell the system every superconcept for any newly added conceptual complex - even a composite concept that should "wear its meaning on its sleeve."

2.1.6 Complex names

Representation schemes that suffer from inadequate definitional facilities tend to hide the internal structure of concepts within their names. We might get the impression that POLYGON-WITH-FOUR-SIDES is a meaningfully structured concept, but in reality it is just as primitive as ELEPHANT. One interesting consequence of the lack of definitional facilities is that the only sense in which these frame systems are "representations" is that their authors have arbitrarily assigned meaning to a set of atomic symbols. There is no notion in a system without definition of representation by structured correspondence. Only if you aren't allowed to lie about properties can the system automatically "know" that a rhombus was a quadrilateral.

2.1.7 Search

We should here take brief note of the effects on search of the design decisions we have been discussing. In a system such as KL-ONE, in which the primary constructor is a definitional or complex-concept constructor, there are enforced limits to search.

For example, search for the instances of a (KL-ONE) Concept can be managed so that it never need go beyond any Concept that is not analytically subsumed by the target Concept. But, in systems in which default-type links are primary, there is always the possibility of a cancel link (or a sequence thereof) and that possibility might necessitate an unconstrained and, in the extreme case, global search of the network. Let us imagine that our target nodes are instances of the Concept THREE-LEGGED ELEPHANT. In KL-ONE, the search would be directed to the target generic Concept (on the basis of its structure) and then would be delimited to searching down through Concepts subsumed by the target, until it gets to the level of individual Concepts that are marked as individuators of those subsumed Concepts. This is the case no matter what else is in the network. In a default-based system, on the other hand, such a delimitation of the search space can not be guaranteed to find all instances of the target, and any strategy based on assuming that some such limits do apply is at best heuristic. For nothing in the semantics of the formalism prevents having nodes for individual THREE-LEGGED-ELEPHANTS under the node for the general concept FOUR-LEGGED-ELEPHANT (with the associated number facet cancelled), nor is it impossible to have nodes for individual FOUR-LEGGED-ELEPHANTS under the generic node THREE-LEGGED-ELEPHANT (again with the number facet cancelled). Indeed, it is not clear what disallows individual THREE-LEGGED-ELEPHANTS under the node for (e.g.) LION-WITH-THREE-LEGS; for though the typical lion surely is not

an elephant (nor an elephant-with-three-legs), perhaps (some) such radically atypical creatures as instances of the concept LION-WITH-THREE-LEGS might also be instances of THREE-LEGGED-ELEPHANT.

Another source of confusion in prototype-based network is that there are several ways to interpret "three-legged" as an adjective:

1. it can be characteristic of the type (as four-leggedness is with elephants)
2. it can be an inherent property of some particular anomalous individual born of a parent of a type with a different characteristic property (e.g., a congenital defect); and
3. it can be an incidental property accruing to some individual born normally of a type with a different characteristic property (e.g., an elephant born with four legs but with an unfortunate experience in the war).

This being the case, we must be careful even when calling a node "THREE-LEGGED-ANIMAL" or "THREE-LEGGED-ELEPHANT" without attaching somehow a causal story of how a so-described creature got that way. If we simply take the normal senses of the two English noun phrases, then it is possible that a three-legged elephant ("normal" sense: type 3) is not a three-legged animal ("normal" sense: type 1).

Note that this is only a problem in a strict prototype-representation, where a concept does not wear its structure on its sleeve. In a carefully specified framework with conceptual

composition, names (for non-primitives) are completely irrelevant; which kind of three-legged elephant we are talking about can be read directly off of the concept's representational structure.

Every composite concept should have a proper place in the network on the basis of its internal structure alone; and one very important job of a representation system is to keep concepts in their places. But no matter how hard we want to believe that the concepts in our representations have intrinsic meaning, unless the system can distinguish between defaults and definitions, "they all look the same" to it.

2.1.8 KL-ONE - Classification

KL-ONE, in contrast to frame-based systems, has focused from the start on the construction of complex concepts, using a small number of structure-building operations on a base of primitive concepts supplied by the user or applications designer. (These operations are described in [11]). One of the central themes of the work on KL-ONE has been that these operators must give rise to complex conceptual entities whose constituent structures are transparent to and usable by the KL-ONE interpreter. This requirement makes possible one of the most significant features of KL-ONE: automatic placement of newly defined conceptual objects in the lattice of concepts. That is, the KL-ONE system, perhaps uniquely among semantic network formalisms, allows a user to generate and extend automatically a lattice of Concepts

induced by the definition of complex Concepts in terms of simpler ones and, correlatively, automatically to place Concepts within this ordering in virtue of their defining constructs (their logical form together with their conceptual constituents). [This automatic classification is independent of the extensions of the Concepts in the actual context of application; indeed, it is independent of their extensions in any possible context. One Concept is a specialization of another even if, for instance, neither has any actual instances - and, in the most extreme case, even if neither has any possible instances.]

To return to our favorite example: the Concept THREE-LEGGED-ELEPHANT (which might be expressed in a standard quantificational formalism extended to allow for the formation of complex predicates as follows: lambda x [(lambda (y) (has y legs) 3) x & elephant x] - see [34]) will be placed as a subconcept of both the Concept ELEPHANT and the Concept THING-WITH-THREE-LEGS (and hence also of THING- WITH-LEGS), and this placement will depend, not on its name, but on the structure and constituents of its definition. (It is to capture this aspect of KL-ONE, among others, that we occasionally have recourse to the language of the extended quantificational calculus -in this case, too, the semantics of such complex predicates as that illustrated above is strictly a function of the structure of its defining term.)

2.1.9 KL-ONE - Inheritance and Inference

Compare this treatment of complex concepts with that afforded in most semantic network formalisms. As noted above, such systems do not allow automatic placement or classification, and essentially what blocks such automatic classification is the decision not to accommodate full-fledged structural composition of concepts. This decision also has the effect of blocking the use of the taxonomic lattice that results from the application of the classifier for realizing certain rules (patterns) of inference that depend on interconceptual dependencies. The simplest example here is that of the rule of conceptual inclusion; the rule that, e.g., any entity (possible or actual) to which the concept *THREE-LEGGED ELEPHANT* applies is also an entity to which the concepts ELEPHANT, THREE-LEGGED, and LEGGED apply. [This rule is the analogue at the intensional level of the combination of the rules of universal instantiation and modus ponens.] Indeed, one can think of the classifier (the program that executes automatic placement or classification of defined, complex concepts) as performing part of the work of a theorem-prover running over formulae of a high-order quantificational formalism extended to include the formation of complex predicative terms via lambda abstraction, in particular, that part of the work which depends on the structural connections between complex lambda terms and between such terms and their primitive, atomic constituents.)

2.1.10 KL-ONE - Decomposition of the SuperC Link

The point about the classifier as a part of a theorem-prover raises another related issue. The SuperC link between Concepts is the structural backbone of KL-ONE. (Again, see [13]; but see also [10].) In conformity with our previous discussion, it must be understood as a complex-concept forming operator (and not, e.g., as a truth-functional connective between sentences). What then, in KL-ONE, plays the role of the material conditional - the "if...then___" connective involved in standard modus ponens: P and if P then Q; therefore Q?

The answer to this question is based on the observation that the relation of conceptual inclusion (the relation between the concepts THREE-LEGGED-ELEPHANT and ELEPHANT, for instance) subsumes (1) the validity of the corresponding conditional ("Necessarily, if something is a three-legged elephant it is an elephant") and thereby (2) the truth of the material conditionals derived by instantiation (e.g., "If Clyde is a three-legged elephant, Clyde is an elephant".) This has led us to decompose the SuperC link so that we may more easily realize standard antecedent and consequent reasoning - that is, reasoning via modus ponens and its "contrapositive" modus tollens (If P then Q and not-Q; therefore not-P). In effect, the link has been decomposed into its purely definitional, structuring component (which is, in turn, understood as implying the validity of the corresponding conditional) and a component that expresses the

standard, universally quantified, material conditional. Such decomposition, then, allows us to distinguish cleanly between the validity or necessity of the conditional and the context-dependent truth of the conditional. (Hence, we can distinguish between conditionals true as a matter of contingent fact and conditionals true in every possible context.)

2.1.11 KL-ONE - Natural-Kind Concepts

We argued above that any representational formalism that aspired toward expressive completeness should accommodate complex, defined concepts. But we also agreed that any such formalism would have to come to grips with the phenomenon of natural-kind concepts - concepts that are best treated as indefinable in terms of any reasonable set of primitives (together with any reasonable collection of conceptual constructors). Decomposition of the SuperC link is one step toward accommodating such concepts in KL-ONE; but we discovered that by itself it was not sufficient. So we have added to KL-ONE a special feature on Concepts, called MAGIC (see the discussion in Section 6.5).

The crucial aspect of this feature is the way that Concepts marked MAGIC (natural-kind concepts) are treated by the classifier. In particular, no complex, defined Concept that does *not* have a given Concept marked MAGIC as a constituent can either subsume or be subsumed by that Concept. Thus, for example, if our network contains the Concept MAMMAL and also contains

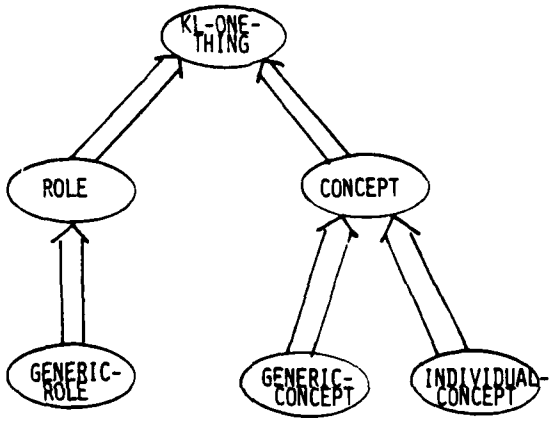
Concepts for some of the mammalian species (e.g., ELEPHANT, DOG, CAT, etc.), then the Concept FOUR-LEGGED MAMMAL will not subsume any of the species Concepts (nor, of course, will it be subsumed by them). This is as it should be, given that the classifier is driven only by full SuperC links - with their definitional, analytic, import - and given that, for example, elephants are not by analytic necessity four-legged. Note that the decomposition of the SuperC link allows us to express the contingent hypothesis that all elephants are four-legged (and are mammals); but note also that nothing in KL-ONE as so far described allows us to handle defaults. A number of suggestions in this regard have been made and are being actively investigated.

2.2 Applications - KL-ONE in KL-ONE

One crucial test of the expressive power of a representational formalism is its ability to describe its own syntax and semantics. This test was forced upon us by the scenario described in Chapter 3. The first task was to describe KL-ONE's characteristic syntactic (graph-theoretic) structures in KL-ONE (and then to describe the way the graphical display facility would present these structures to the user).

We should add that we shall not be describing a mechanism for automatically generating meta-descriptions of KL-ONE in KL-ONE. Such a facility is desirable, but its design and implementation are tasks requiring much further work.

We begin with the bare bones of the taxonomic lattice of KL-ONE objects. [See Figure 2. The sentences below the figure are JARGON sentences which express most, but not all, the information embodied in the diagram.]

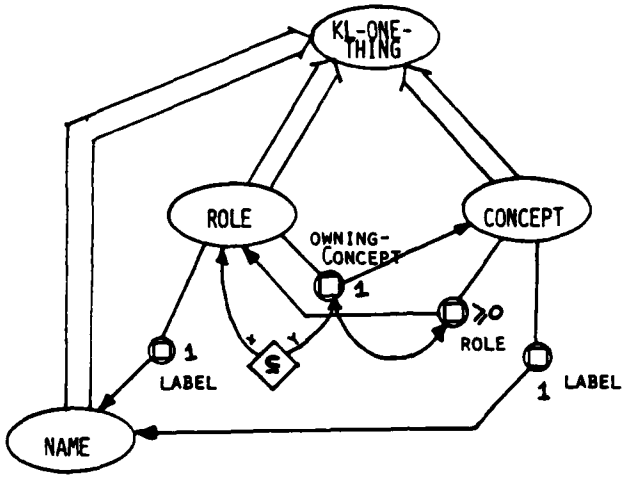


A ROLE IS A KL-ONE-THING.
A CONCEPT IS A KL-ONE-THING.
A GENERIC-ROLE IS A ROLE.
A GENERIC-CONCEPT IS A CONCEPT.
AN INDIVIDUAL-CONCEPT IS A CONCEPT.

FIG. 2. THE HIERARCHY OF KL-ONE OBJECTS.

As shown, a crucial top-level dichotomy is that between ROLES and CONCEPTS, and in Figure 3, we express the main facts about the relationships between these two major syntactic types. [This can be confusing. The Concept ROLE is itself not a Role but a

Concept; the Concept CONCEPT has a Role (named **role**) that must be filled by individuals of the type ROLE. ("A CONCEPT HAS PL ROLE WHICH ARE PL ROLE.")



A ROLE HAS A LABEL WHICH IS A NAME.
A NAME IS A KL-ONE-THING.
A CONCEPT HAS A LABEL WHICH IS A NAME.
A ROLE HAS AN OWNING-CONCEPT WHICH IS A CONCEPT.
A CONCEPT HAS PL ROLE WHICH ARE PL ROLE.
A ROLE IS A SUBSET OF ITS OWNING-CONCEPT'S PL ROLE.

FIG. 3. THE RELATIONSHIP BETWEEN CONCEPTS AND ROLES

Each Role belongs to a particular Concept - hence the Concept ROLE has a Role (named **owning-concept**) that must be filled by a Concept (an individual of type CONCEPT).]

Figure 4 expresses in more detail the structural relationships between Roles and Concepts, and the following

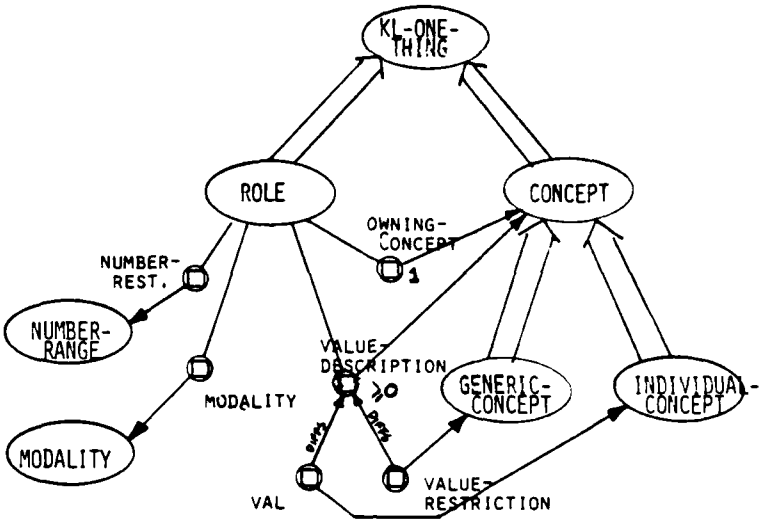
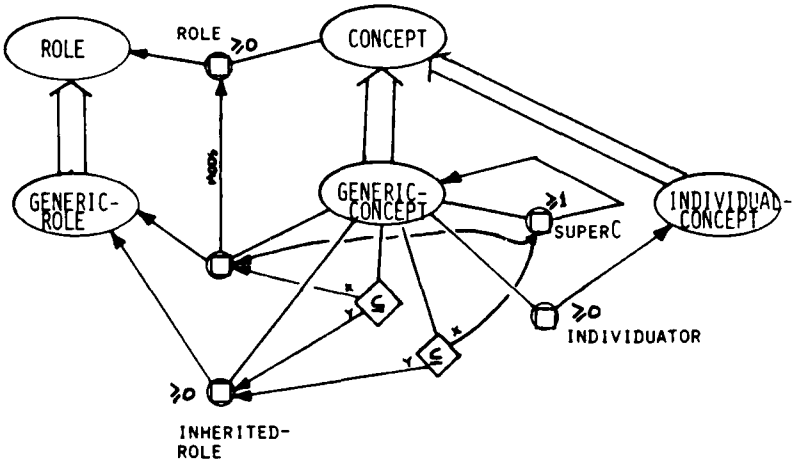


FIG. 4. MORE ROLE STRUCTURE

diagram (Figure 5) fills out more of the definition of GENERIC-CONCEPTS as a syntactic type. All of the information so far has been at a completely general level - in the next figure, Figure 6, finally, particular generic Concepts are introduced. Note that in this description of, for example, the generic Concept: PERSON, the Concept is an individual of the type GENERIC-CONCEPT; hence it is associated with an individual Concept that is an individuator of the generic Concept GENERIC-CONCEPT. (Just as in



A GENERIC-CONCEPT HAS PL SUPERC WHICH ARE PL GENERIC-CONCEPT.

A GENERIC-CONCEPT HAS PL INDIVIDUATOR WHICH ARE PL INDIVIDUAL-CONCEPT.

THE PL ROLE OF A GENERIC-CONCEPT ARE PL GENERIC-ROLE.

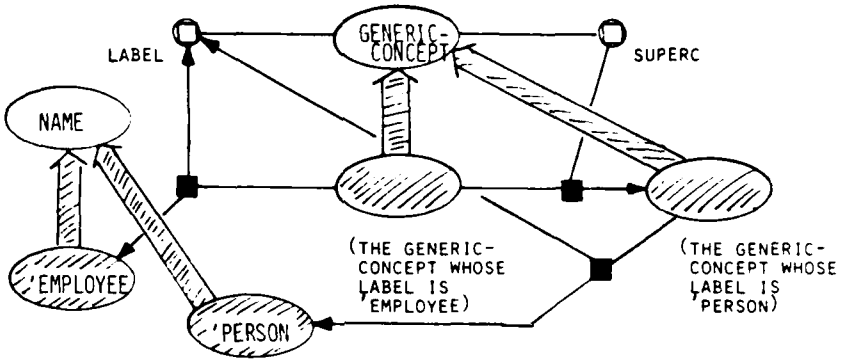
A GENERIC-CONCEPT HAS PL INHERITED-ROLE WHICH ARE PL GENERIC-ROLE.

THE ROLE OF A GENERIC-CONCEPT ARE A SUBSET OF ITS INHERITED-ROLE.

THE ROLE OF THE SUPERC OF A GENERIC-CONCEPT ARE A SUBSET OF ITS INHERITED-ROLE.

FIG. 5. MORE ABOUT GENERIC CONCEPTS

a description of the syntax of English, the word "person" can be described as a common noun - as an instance of the type: common noun, and "common noun" itself is a common noun.) This figure also expresses the SuperC connection between the concept PERSON and the Concept EMPLOYEE. It does this by way of the superc Role on the Concept GENERIC-CONCEPT, marking that every generic Concept has at least one Concept which subsumes it. [By a slight



'PERSON IS A NAME.
'EMPLOYEE IS A NAME.
THE SUPERC OF THE GENERIC-CONCEPT WHOSE
LABEL IS 'EMPLOYEE IS THE GENERIC-CONCEPT
WHOSE LABEL IS 'PERSON.

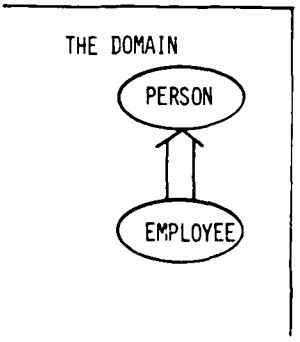


FIG. 6. SOME PARTICULAR GENERIC CONCEPTS

extension, the highest Concept in a taxonomy, or indeed the Concept THING or ENTITY which is the highest allowed, can be said to subsume itself and to be subsumed by no other Concept.]

The next and last figures (Figures 7 and 8) in this series show some of the structures associated with the scenario, in particular with interchange number [9]. The large figure (Figure 7 describes the KL-ONE structure shown in the small figure (Figure 8).

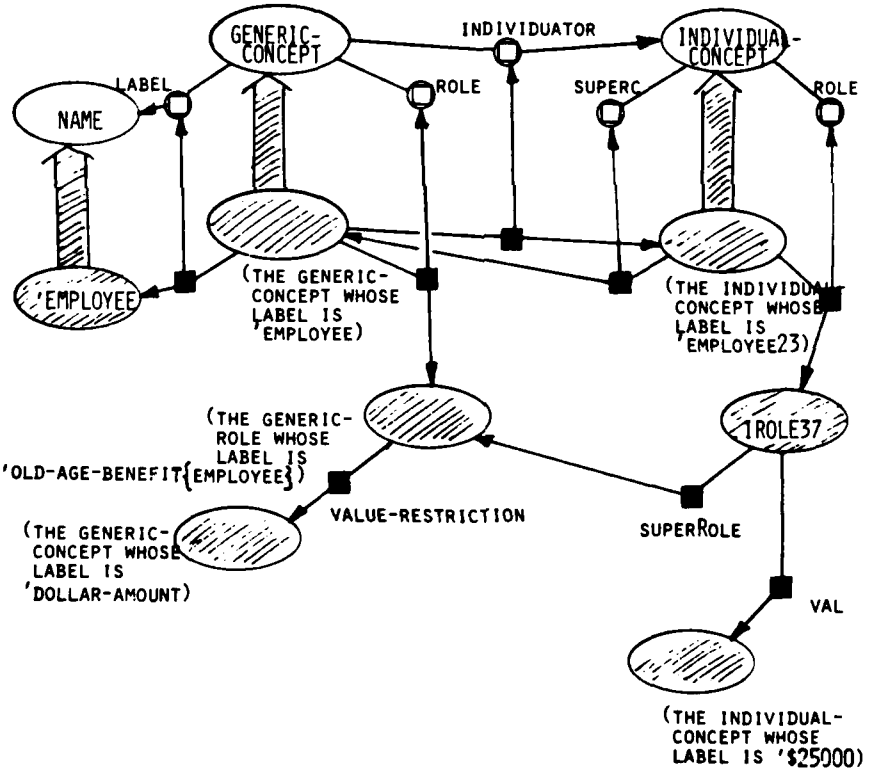


FIG. 7. RELATIONSHIPS BETWEEN INDIVIDUAL AND GENERIC CONCEPTS

The main figure, as complex as it is, is in many respects highly

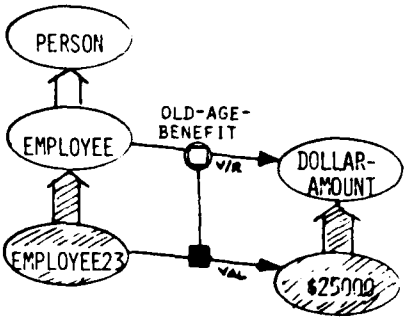


FIG. 8. THE FRAGMENT DESCRIBED IN FIGURE 7.

simplified; the crucial simplification lies in the fact that, as shown, the generic Concept EMPLOYEE lacks interesting structure. (Indeed, as shown in that figure, it is not even marked as a Subconcept of PERSON; but see Figure 8). Note here that the individual Concept EMPLOYEE23 is itself marked as an individual Concept and hence is an individuator of the generic Concept INDIVIDUAL-CONCEPT; but it is also marked as being a filler of the Role individuator with respect to the generic Concept EMPLOYEE. An analogous situation holds with respect to the structural relations between the individual Role irole37 and its SuperRole old-age-benefit-of-employee.

2.3 KL-ONE Implementation

by J. Schmolze and R. J. Brachman

An implementation of the KL-ONE language has been under

continuous development at BBN for several years. This past year's implementation centered on extensions to the KL-ONE system, transfer to the Jericho machine and development of KL-ONE software packages.

2.3.1 Changes to the KL-ONE system

The KL-ONE system has grown in several useful ways during this past year. Primarily, we overhauled the programmer's interface to KL-ONE (i.e., the set of Lisp functions within KL-ONE that an applications programmer may use). This overhaul included software modifications and extensive documentation. In particular:

- The KL-ONE program has become quite large, with several programmers working on it, and has users at BBN and ISI. Early last year, we reorganized the KL-ONE program and distributed its software onto 12 files. This partitioning allowed several programmers to work on KL-ONE simultaneously. We introduced controls for access to our software to facilitate its distribution among both programmers and users. All users of KL-ONE can access software that is ready for distribution, but only KL-ONE programmers can access software still being developed. These controls were accompanied by a numbering scheme for releases of KL-ONE to aid the bookkeeping associated with releasing and maintaining software.

- The theoretical design of the KL-ONE language is continuously evolving, forcing us to update the KL-ONE program periodically. There were several of these updates this year. For example, we discarded the notion that Concepts reside in Contexts; later, we changed the KL-ONE program so that functions creating Concepts no longer have an argument for Context. Other changes show our increased awareness of the distinctions between differentiation and modification for Rolesets, reflected in updates to our Roleset inheritance algorithms.

- Some edits were made with regard to programming

conventions, such as the removal of the use of "wiring lists". They had provided a notational shorthand for describing the Roles of a Concept but went largely unused because of their highly specialized syntax. The CKLONE package, written by Frank Zdybel at BBN, performs the same function with a simpler and more natural syntax.

- We also added functions that were conspicuously missing from the user interface. Some of these were information retrieval functions that examine only local structure, thereby ignoring inheritance. These functions allow a user of KL-ONE to view the actual network topology instead of the one implied by inheritance. A need for this procedure arose from the AIPS group at BBN. They constructed an object-oriented programming schema for KL-ONE Concepts that needed its own variant of inheritance for certain properties associated with Concepts (these properties contained the procedural definitions that would execute messages sent to Concepts). In order to define their own inheritance scheme, they required local network information.

- The most important product of our overhaul, however, was the update of documentation for the user interface functions, as much of it had fallen out of date. Some time ago, we wrote a tool to generate documentation from the actual function definitions. It combines the comments in each function with information it can extract from the structure of the definition. Using this scheme, each function contains, and supplies, its own documentation. The tool was extended to create a file suitable for a text formatter whose final form is a manual of KL-ONE user-interface functions [14]. We have distributed these documents to the users of our software.

2.3.2 Transfer to Jericho machine

Last spring, we successfully transferred the KL-ONE program to BBN's Jericho machine. Since then, all KL-ONE software work has been done exclusively on Jerichos. Other than the simple problems caused by the difference in the directory structure between Jericho and TOPS-20, we had difficulty only in transferring the DECL Lisp-user's package, which KL-ONE depends upon. Once this was completed, KL-ONE came up readily.

In an attempt to judge long range limitations of KL-ONE on Jericho, we analyzed the space required for storing Concepts and Roles with the hope of estimating the maximum number of Concepts that could fit into the address space of Jericho. We tried to estimate the size of "average" Concepts, where we grant that the meaning of "average" is vague. Of course there are many parameters for this estimate. For example, we chose the average number of modified roles for a concept to be 3, with the total number of local roles being 4. After taking all parameters into account, we calculated the space required for each "average" concept, which was 860 bytes on Jericho. The results are summarized in the following table.

TABLE 1. *ESTIMATED NUMBER OF CONCEPTS THAT FIT IN AN ADDRESS SPACE*

Type of machine	DEC-20/ TOPS-20	small Jericho (16 mega-bytes)	large Jericho (64 mega-bytes)
# concepts	300	14,000	70,000

Note that in the above table, the estimate for TOPS-20 comes from the experience of KL-ONE users. For Jericho, the estimates are from the calculations described above and allow for programs, including KL-ONE, to occupy 4 mega-bytes.

2.3.3 KL-ONE software packages

Whenever possible, we have designed extensions to our KL-ONE implementation to be separate packages that can be included at

the discretion of each user, thus allowing users to ignore extensions they do not need. One package under consideration will write a KL-ONE network onto a text file that can be read into a different system to create a copy of the original network. This will be a new input/output package for KL-ONE networks, replacing an existing package that has fallen out of date and out of use. We are still in the initial design phase, with contributions coming from BBN, Burroughs, Fairchild AI Lab, ISI and Xerox PARC. We intend the notation we choose for representing networks on text files to be:

- machine independent, so we can share networks between sites, as KL-ONE runs on three types of machines (TOPS-20, Jericho and Dorado/Dolphin) and will soon run on VAX machines under FranzLisp;
- easily readable, similar to the notation for Lisp functions that a prettyprinter generates;
- suitable for the integration of one network with another.

Another package that is almost ready for inclusion in KL-ONE is a classifier, constructed by the CONSUL group at ISI. The classifier uses the fact that relationships between KL-ONE Concepts have the status of definitions and places a Concept at its "appropriate" location in a network. The "appropriate" location is below all Concepts which subsume it (Subconcept) and above those which it subsumes (Superconcept). The classifier already works; we need only design its interface to the rest of the KL-ONE system.

2.4 Interactions with Other Research Groups

Over the last several years, BBN's Knowledge Representation for Natural Language Understanding project has had a significant impact on the field of Artificial Intelligence. A number of groups outside of BBN are now either using one of our implemented subsystems, planning to use one, or basing their further work on our ideas - in some cases using paper and pencil versions of our frameworks.

2.4.1 First KL-ONE workshop - 1980

The most recent evidence of the extent to which our work has permeated the area of knowledge representation was the attendance at the KL-ONE Workshop, held last October. Gathered in Jackson, New Hampshire, were 49 people with serious interest in using KL-ONE in some capacity, or with significant experience in using it already. Most important, that group represented about 21 institutions other than BBN: The Wharton School, University of Rochester, NOSC, WPI, Simon Fraser, Harvard, University of Illinois, University of Delaware, University of Massachusetts, University of Amsterdam, DARPA, NIH, Xerox PARC, SUNY/Buffalo, University of Pennsylvania, Burroughs, GMR, Boston College, ISI, MIT and TI. In addition, we had attendees from two other groups at BBN.

At the workshop, our research group reported on recent progress and plans for future investigations. We requested, and

received, feedback about KL-ONE as a general-purpose representation language: who is using it or plans to use it; how they use or plan to use it; what is wrong with or missing from KL-ONE. A popular request was to translate KL-ONE to a Lisp that runs on the VAX machine, as the VAX is more common than machines that already run INTERLISP. Efforts in this regard are reported later in this section. We held several small discussion groups concerned with (1) particular representation problems, such as representation of beliefs, and (2) implementation topics, such as the possibility of a virtual KL-ONE language. However, the primary advantage of the workshop was the gathering of these researchers who use KL-ONE to meet with each other and discuss their projects and problems.

2.4.2 Primary collaborations

Over the past year, the most experienced users of KL-ONE and the KL-ONE designers at BBN have developed a strong collaborative effort. Much discussion has occurred concerning important issues facing KL-ONE's design, such as: introducing "qua" concepts into KL-ONE; activating Roleset Relations so that they cause a form of inheritance; distinguishing various uses of individuals; plus a host of other topics. Our primary communication medium is the ARPANET, with U.S. mail providing a medium for pictures. The result has been a stimulating correspondence between researchers at BBN, Burroughs, Fairchild AI Lab, ISI, and Xerox PARC. In fact, we met for several days before the 1981 KL-ONE Workshop,

held in October, to have intensive discussions and make plans concerning the direction of our collaborative research for the next year.

These interactions have also led to collaborative software development. For example, ISI built a classifier for KL-ONE that will automatically place new Concepts in a KL-ONE network at their "proper" location[1]. Originally, BBN constructed a classifier as part of the JARGON system. However the ISI version will supersede, as it is more complete than the BBN version. Additionally, small software fixes and additions have been made by ISI and other groups at BBN who use KL-ONE, and BBN has shared these with the remainder of the community.

2.4.3 Translation of KL-ONE to FranzLisp for VAX machines

In May 1981, Tim Finin at the University of Pennsylvania approached us with the hope of constructing a translator to take our Interlisp software for KL-ONE into FranzLisp for use on VAX machines. Since that time, the translator has been built, KL-ONE has been translated into FranzLisp, and the FranzLisp version is being tested and debugged. Tim Finin and others performed the bulk of the work for this task, with assistance from BBN, CCA and Temple University. BBN's contribution was to provide:

---

[1] This classifier is described in slightly more detail in the section on KL-ONE Implementation 2.3.3.

- Consultation for the design of the translator
- Interlisp expertise and information concerning the precise definitions and arguments for Interlisp functions (in particular, for functions used by KL-ONE)
- Source code for KL-ONE
- Assistance with the translation by preprocessing each file to rid usage of CLISP.

The current estimate for release of a FranzLisp version of KL-ONE is January 1982. At the 1981 KL-ONE Workshop, we met with prospective users of this system to arrange protocols for software distribution and maintenance.

2.4.4 Other collaborations

Besides the KL-ONE Workshop, we have had a history of collaboration, support, and impact on the following research labs and university Computer Science departments:

- Universities:
  - University of Illinois: two graduate students are using KL-ONE in their dissertation research work.
  - University of Pennsylvania: one faculty member's recent thesis on nominal compounds used a representation framework substantially influenced by KL-ONE; other faculty are doing work on discourse-level phenomena based on work done in the past on our project; other faculty are translating KL-ONE software from Interlisp to FranzLisp to run on VAX machines; other faculty implemented a small version of KL-ONE in PROLOG.
  - MIT: various graduate students have at one time or other worked on our projects; one undergraduate thesis is based on work done here; a recently completed Ph.D. thesis on language generation used KL-ONE as one of its representation languages.

- Harvard: an undergraduate thesis has been completed on marker-passing simulator built to support inference in KL-ONE networks; an almost-completed Ph.D. thesis was written explicitly as an extension to KL-ONE work. Another thesis using ATN's may be extended in future work using the KL-ONE system.

Research Laboratories:

- Sperry-Univac: a research group is using RUS as a basis for a front-end to a database system. They are considering usage of PSI-KL-ONE to create and process semantic representations of queries in KL-ONE.

- Burroughs: there is work on extensions to KL-ONE to handle events and histories, "qua" links, etc.; and a proposed collaborative project to develop KL-ONE implementation in PLAN.

- Xerox-PARC: implementation of KL-ONE in SmallTalk (KloneTalk) is under active development and extensively used.

- ISI: full-scale collaboration is providing KL-ONE and RUS parser implementations; both used actively as integral part of Consul system.

- CCA: they are planning to use KL-ONE structures in VIEW, the successor to spatial data management system.

- Other groups at BBN include the AIPS project, which uses KL-ONE as integral part; and interaction with Brian Smith on MANTIQ. [63]

3. OVERVIEW OF THE NATURAL LANGUAGE SYSTEM

3.1 Goals for a Natural Language System

One of the goals of the Knowledge Representation and Natural Language group at BBN has been to provide powerful general tools for natural language processing to build language understanding systems for a decision maker using a graphics display. We have in mind decision makers accessing information from a database that can be represented visually; they need to collect information from the database, add to it, change it, and define new features. A special feature of this language understanding system is the assumption that human users express themselves naturally. They can utter more than direct imperatives and can ask questions other than the direct questions typical of most current AI language systems.

As a first step in building a natural language system that meets these goals, we researched, designed, and built an experimental prototype that accepted natural English from a user to generate ATN networks on a two-dimensional display. Our current effort is aimed at a system with more linguistic sophistication than our first prototype. We have planned for expanded capabilities in several dimensions. These include the capacity to interpret many different kinds of referential phrases, as well as the intended meanings of new sentence types (such as reports of errors, direct information retrieval

questions, requests for action via questions, multi-utterance requests, elliptical questions, clarifications, reports of plan changes, and reports of new plans and additional task capabilities (such as recognition of errors in the speaker's plan and use of expectations from recognizing a plan). We believe these capabilities are basic to a system that will serve a decision maker's needs in a command and control situation.

In this chapter we outline the kinds of domains we have been exploring for prototype systems. We also report on an investigation by Bobrow, Sidner and Webber of the tasks typical in a newly chosen domain, and then illustrate a sample interaction between a user and the proposed prototype system for this domain.

3.2 Domains for Experimentation

by C. L. Sidner

In our first ARPA proposal [77] (for research on natural language) we proposed to investigate techniques for a system which could deal with dialogues like the one below.

1. Cdr: Show me a display of the eastern Mediterranean.
   [computer produces display]
2. Cdr: Focus in more on Israel, and Jordan.
   [computer does so]
3. Cdr: Not that much; I want to be able to see Port Said and the island of Cyprus.

[computer changes scale and window to include the desired features]

4. Cdr: Now show me the positions of all U.S. and Soviet vessels in the area.

[computer does so, and makes a default assumption for displaying the difference between U.S. and Soviet vessels]

5. Cdr: Where is the John F. Kennedy?

Computer: Two hundred miles to the west of the point displayed.

[The ship is not on the screen, so the system displays a point at the left edge of the display]

6. Cdr: Show me the course tracks for the Soviet vessels for the last five hours.

[computer does so]

7. Cdr: What kind of ship is that?

[points to a Soviet vessel]

Computer: Soviet missile cruiser.

8. Cdr: Show me the other missile cruisers, and display all vessel types with two digit code.

[computer blinks or flashes all of the missile cruisers for 2 1/2 seconds and displays with each vessel the two digit type code (assumed previously agreed on by the commander)]

9. Cdr: Remove the course tracks, and show small dots with one-hour course tracks for any known Soviet aircraft in the area.

[computer does so]

[commander makes his assessment of the situation and makes appropriate orders for his forces]

10. Cdr: Remove the planes and track the Soviet vessels for the next four hours. Show any deviations from current course double intensity and ring bell when detecting course change. Flash vessel changing course for 10 seconds.

[computer accepts standing orders for continual monitoring and conditional future behavior]

To explore some of the problems implicit in this scenario and to develop some techniques for dealing with them, we developed a prototype based in an ATN domain. Our scenarios for this prototype (see [13] for the user's utterances and the system's actual response) included direct imperatives to be taken as direct commands, as well as declaratives to be taken as indirect commands (as in exchanges 1 through 4 in the above scenario).

More recently we conducted an experiment in collecting protocols of users interacting with simulated versions of the system we envision. Our analysis of those protocols convinced us that the behavior exhibited in exchange 3 above is the "tip of the iceberg" of a much more varied and common linguistic use. In particular, people often discuss a wide variety of changes they require in a computer system for reasons due both to changing their minds and to misunderstandings of what the system can or would do. Such users negotiate making changes in particular ways, and they comment on the system's progress as the changes occur. To provide a context in which to develop techniques for modeling such behavior, we have selected an additional domain where the graphic display problems are similar to real command and control situations, and where the kinds of exchanges users have with a system mirror the behaviors described above. We call

this the KL-ONE-ED domain. In the next section we discuss the types of user tasks and system operation to be found in that domain.

3.3 The KL-ONE-ED System

by R. J. Bobrow, C. L. Sidner and B. L. Webber

There are two classes of tasks to which we expect the KL-ONE-ED system to be put: that of a layout assistant to enable the user to design interactively a maximally effective display of some known or new KL-ONE structure as a text illustration) and that of a graphic editor to enable the user to construct, manipulate, or modify an otherwise difficult to comprehend data structure.)

These two uses vary not just in their intent and effect but also in the common sequences of utterances a user would make to enlist and manage the system's help. Moreover, whereas similar utterances might be made in both types of task interactions, the system response that would be appropriate in each case might be very different. For example, consider the utterance "Remove this concept." Within a layout task, it may be the concept's suppression on the virtual screen that the user wants, whereas within an editing task, it may be that concept's removal from the current virtual data base (i.e., from the current overlay) that is intended. (See the Glossary, Section 3.3.3 for brief descriptions of these and other boldface terms.)

Another example involves an utterance like "Show me the EMPLOYEE concepts". In layout, the intended effect would probably be their simultaneous appearance on the screen, placed according to the current presentational conventions. For editing purposes, though, users might expect to have them shown one by one, so that they could make appropriate changes to one before going on to another.

Now, while users involved in a layout task will primarily be interested in operations that affect the virtual screen and their window onto it, and while users involved in editing are primarily interested in operations that affect one or more virtual data bases, both types of operations will clearly be needed in satisfying both types of task. For example, in editing, users may attempt to get more room on the screen (a layout task) before adding in a new concept, while in layout, they may be both creating and modifying a new data structure as well as arranging it on the screen.

Notice that task and subtask are being used here to refer to things a user wants done, be they "multi-step" desires like "Create a partition of the Generic Concept DOG" or single-step ones like "Scratch the IConcept FIDO." Operation, on the other hand, is being used to refer to things the system can do, like suppressing a data base object or moving the window around the virtual screen. From the point of view of natural language interaction, a user can explicitly request the system to perform

an operation - so that, in some sense, a task can be fulfilled by a single operation - or the system can **ask** the user whether he or she wants a particular operation performed (see exchange [6S] in the scenario in Section 3.4 where the system responds to the user with "Shall I empty the screen and save the current display?")

The kinds of user tasks and system operations with which we will be concerned for layout and graphic editing are the following:

General Facilities

- o Constructing and/or modifying a general data base
- o Constructing and/or modifying a presentation
- o Constructing and/or modifying a KL-ONE virtual lattice

3.3.1 User tasks and system operations

3.3.1.1 Task: Identify System Capabilities and/or Contents

This task requires the user's ability to query and thereby learn about any of the various knowledge sources - the current virtual data base, the system's meta-knowledge, the current set of presentational conventions, or the current standing orders. Subtasks that the user might be involved in include:

- o Finding out some characteristic of a Concept before displaying it - e.g., whether there's likely to be enough room to display all its SubConcepts, either its immediate ones or all of them,
- o Finding out whether certain Concepts exist before asking for them to be displayed,

- finding out system capabilities.

Relevant system operations are those needed for any cooperative question-answering system able, among other things, to

- Carry on either a linguistic, graphic or mixed-mode interaction with a user
- Correct the user's misconceptions
- Enter into a clarification or negotiation subdialogue.

3.3.1.2 User Method: Try out Multiple Alternatives

This method is useful both in preparing layouts and in incrementally developing a desirable change or addition to the data base. Relevant system operations are those needed to:

- Develop a new virtual data base and/or presentational conventions for producing a virtual screen. The specific operations involved in these tasks will of course vary with the type of data base. For a KL-ONE type data base, the operations are discussed in the section 3.3.1.4 below.
- Manage a set of data base overlays. This process involves directly or indirectly invoking system operations to:
  - Name an overlay.
  - Flip back and forth between overlays
  - Assimilate an overlay into the data base to produce a new real data base.
- Manage a set of presentations. This process involves directly or indirectly invoking system operations to:
  - Name a presentation.
  - Flip back and forth between presentations.

3.3.1.3 Task: Create a Desired View of Data

Except for those situations in which the user wants to see objects that are not in the real data base, subtasks here involve changing the display via alterations in the presentational conventions and not via changes to the virtual data base. (Of course, the creation of new objects or alterations to old ones, even if intended just for display purposes, implies that the current virtual data base is no longer equivalent to the old one. Provided that the old one is not later "frozen in," there is no problem.) The following list includes subtasks that the user may have to perform:

- Getting a set of objects on the screen. This subtask may involve:
  - Constructing some new objects - see discussion in 3.3.1.4
  - Focusing on a region of the virtual data base (or, in other words, placing a window over part of the virtual screen. This may involve some computation, in order to assign positions on the virtual screen to relevant objects in the virtual data base.)
  - Zooming or panning around the current area of the virtual screen. This will only increase the number of objects on the screen if both the virtual data base and presentational conventions are such that other data base objects have assigned positions outside of, but "next to", the area currently in view.
- Getting more room to display a particular structure. This subtask may involve:
  - suppressing some details
  - moving the current structure around on the virtual screen

- panning the window around or zooming it out
- zooming the window in
- reducing line crossings
- making an object, set of objects, or substructure more prominent
- achieving a particular correspondence between screen positions of various objects and directions of connections between them.

o Relevant system operations are those needed to:

- establish a window on the virtual screen, using system's default mapping constraints or whatever user-specified constraints currently in force
- rearrange objects on the screen
- reduce objects visible on current and/or future screens
- highlight objects on the screen
- change symbology (size, shape, etc.) used to represent objects, including connections, currently on screen
- change screen captions
- change global scale.

3.3.1.4 Task: Construct or Modify a KL-ONE Data Base

This task requires the user's ability to:

o Differentially subcategorize a concept

- based on "diffing" a particular Role - the user is likely to specify how subcategories differ vis-a-vis that Role,
- based on additional restrictions to V/R,
- based on adding roles - the user is then likely to elaborate each new subcategory in turn,

- based on intersections with subcategories of another Concept - since this leads to a known set of graphical problems (line crossings), the user is likely to attend to this layout task sometime later,
- based on subcategorization of a Role's V/R.

Notice that the first two types of subcategorizations involve adding new information to the lattice. The latter three involve making explicit information already implicit in virtual lattice, hence user must have some additional reason for making these subcategories explicit. For example, the subcategories may also differ by virtue of having additional roles that the user may then mention.

- o Create a partition of a Concept (i.e., subcategorize it both mutually exclusively and exhaustively). This also sets up a context for interpreting subsequent remarks.
- o Draw up one or more abstractions of a set of Concepts.
- o Replace one subcategory of a Concept with a set of others.
- o Change names, classes of names, associated with Concepts or Roles.
- o Add/delete Individual Concepts or Roles.
- o Undo previous changes.

3.3.2 User methods for layout and graphic editing

Users are expected to have certain general methods they might follow in carrying out any layout or graphic editing task. How these methods might be exploited to improve the system's behavior is not immediately clear to us, but it seems useful to keep them in mind. A list of these methods follows.

- o Remove detail from screen before beginning task, in order to make things more visible; put details back afterwards

- o Try out alternative layouts or graph structures - hence, the need to keep several named alternatives for viewing
- o Try alternative ways of achieving task - backtrack whenever stuck and try something else. Note that the more trial and error involved in achieving goal, the less recognizable is the plan, though not necessarily the goal
- o Make a change in one representation, display it in another to see its ramifications, then return to the first for further changes (a la editor/formatter pairs) - in this domain, the user might make changes graphically, then pretty-print them.

3.3.2.1 General System Principles

There are two general principles that the system could adhere to in its interaction with the user, which would reduce the amount of effort it needed to expend, while still behaving satisfactorily. These principles, similar to Sacerdoti's planning principles followed in his NOAH system: "Use existing objects," "Don't order operations before being forced to," etc., are:

1. Make minimal changes to the screen. For example, the system might choose to answer a request like "What are the Roles under EMPLOYEE BENEFITS?" linguistically instead of graphically. If users actually wanted one or more roles put on the screen, they could then ask explicitly.
2. Describe things to the user in terms of results, not in terms of the operations that achieve them. The suggestion made earlier about showing the user ways of getting more room on the screen follows this principle.

3.3.3 Glossary

real data base (or just "data base") Starting point of information on-line. The data base cannot be changed directly, but rather through the

mechanism of "freezing in" or assimilating one or more overlays.

data base overlay
(or just "overlay") Nameable collection of incrementally acquired changes to the data base. The data base isn't actually changed until one or more of these overlays is frozen in. This may involve, a la EMACS, asking users whether they want to name the current overlay or freeze it into the data base whenever they indicate a desire to start afresh, or to start something else (i.e., a non-undoable sort of thing), or log out.

virtual data base
(or "VDB") Real data base plus overlay(s). By flipping back and forth between overlays, the user can see alternative versions of the real data base. One special virtual data base is the real data base plus the null overlay.

active region That portion of the virtual data base, often more than what's currently visible, on which the user would consider himself or herself to be working. In a relational data base, this might be a relation, not just those of its records and fields currently on display. In a data base of faces, the active region might be a face or even a set of faces, even though the window may have been zoomed in to the tip of a single nose. For a KL-ONE lattice, the notion of active region is somewhat more problematic, as there are few if any natural units dividing up the data base. Nevertheless, the notion seems valuable in delimiting an appropriate context for interpreting definite references and quantifiers in the user's utterances. Suggestion: if this turns out to be the only value in the notion, then we may be able to get by with equating the active region of the KL-ONE data base, while the user is involved with a particular overlay, with the set of structures visible or mentioned.

virtual screen (or "layout") Result of mapping virtual data base onto a limitless but scaled 2-dimensional surface. For computational efficiency, not every object in the virtual data base need be automatically assigned a position on the current virtual screen. One strategy open to us is to compute an object's **assigned position** only when it becomes part of the **active region**.

window Conforming to standard display use, that part of the virtual screen currently visible. A window can be panned or zoomed, thereby changing what sanctioned parts of the virtual screen are visible. For now, we will be assuming that the system has a single window. If we decide to take advantage of split-screening, users will be able to declare multiple windows. If these windows are to permit different scale views of the virtual data base, we must consider the problem of what scaling is associated with - a window or the virtual screen.

presentational conventions
Specific and general constraints that the above mapping must satisfy. These may be constraints relating to individual data base objects, sets of data base objects, or classes of data base objects. They include:

- o constraints on **how** data base objects are to be displayed on the virtual screen(shape, size, labeling conventions, flashing, etc.)
- o constraints on **where** objects are to be positioned on the virtual screen.
- o constraints on **what** objects or details are to be **suppressed** (not permitted to be seen) and which ones **sanctioned** (permitted to be seen). N.B., Some suppressed objects and details may have an assigned position on the virtual display, while others do not. Those with an assigned position will not be visible, even when that position is within the window. However, in that situation, if its status changes to "sanctioned," such an object or detail will suddenly appear on the screen. Note - there may be a need to negotiate if the user attempts to put another object "on top" of a suppressed one. This implies that the graphic system should be able to recognize such an attempt.
- o specifications of additional labels and captions not directly correlated with data base objects.

Note that the system is assumed to have a set of default presentational conventions that are used whenever the user has not specified what conventions are to be in force.

presentation The set of commands that produces the virtual screen. We are assuming the ability to manage a set of alternative presentations as well as a set of alternative overlays.

meta-knowledge (or "data base model") Information about what can be in the data base and what the system can do. Meta-knowledge can be used for answering users' queries, but it cannot be changed by the user.

3.4 A Scenario for interacting with KL-ONE-ED

The following scenario is one that the Knowledge Representation and Natural Language Group had chosen to use as its test for our current system design. In the text after each exchange, are comments about the kind of behavior exhibited by the user and the system. We plan to design our system to have the general capacities to respond to users' behavior in ways similar to those shown.

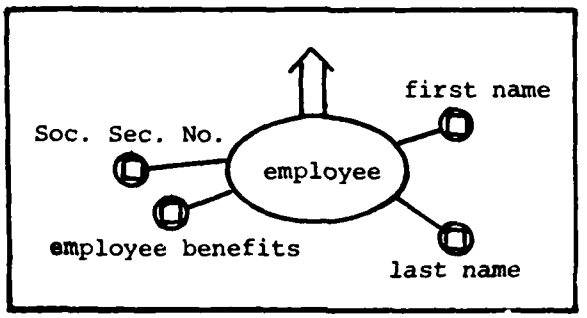
3.4.1 The Scenario

[1]
P: Show me the generic concept
called "employee."

S: OK.

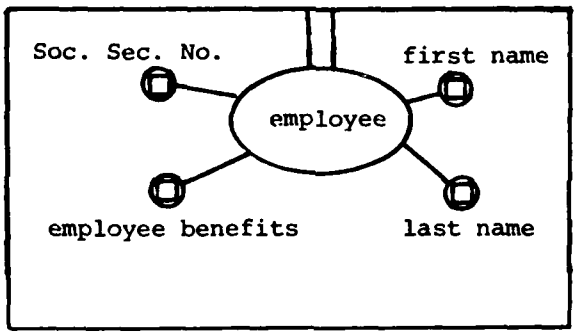
Behavior: standard imperative request and response.

[2]



P: I can't fit a new individual concept below it. Can you move it up?

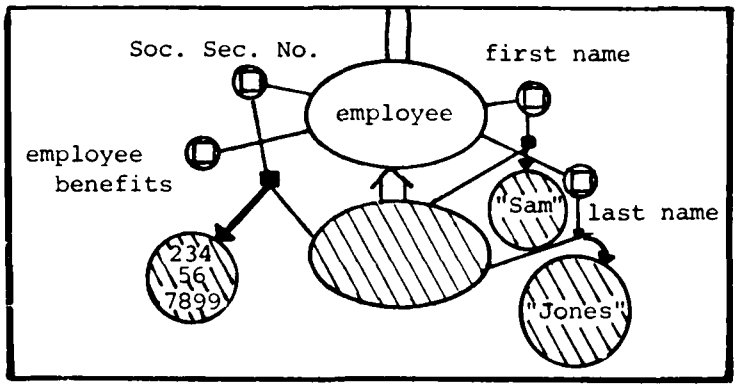
S: Yes.



Behavior: declaratives used to impart information to other participants, questions as used to request action or information, depending on what the user knows.

[3]
P: Ok, now make an individual employee concept whose first name is "Sam" and whose last name is "Jones." The Social Security number is 234-56-7899.

S: OK.



Behavior: a declarative that implicitly indicates where the concept should be placed, and a declarative (2nd sentence) that is also a request.

[4]
P: Is there a role on
employee called "retirement fund"
or something like that?

Behavior: information retrieval question

S: No there isn't. What info
are you trying to add?

Behavior: standard answer plus attempt to get more information about what user is doing. Note that system that user is adding to assumes the display because of the context so far.

[5]
P: How about a role called "pension program" or "pension plan?"

Behavior: Mixed initiative - user doesn't want to answer question because he's trying to find out something he needs to know.

S: Neither is a role. Would you like to see all the roles on employee?

Behavior: question is a helpful response. Note that an imperative form ("Try looking at all the roles on employee") implies that system knows what user is doing, while the question is helpful even though the system isn't sure of exactly what the user wants.

[6]
P: No, I'd like to see the structure below employee benefits.

Behavior: A request that states simply the effect, not how it is that the system is to accomplish it.

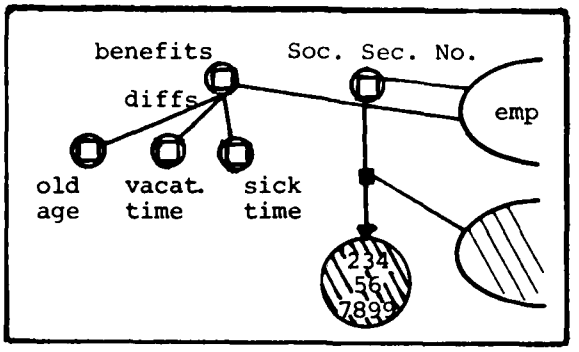
S: Shall I empty the screen and save the current display?

Behavior: System has two possible responses, each with different effects. Asks which is intended by asking about one of them (system here must know that user is aware that it is capable of adding to the display).

[7]

P: No, I want the stuff here <points to screen>.

S:



Behavior: A typical case of pointing, accompanied by reference using a lexical generalization ("the stuff").

[8]
P: What's the vr on old age benefits?

Behavior Typical information retrieval question, but "old age benefits" refers to those now visible from the previous action; a case of referring to visually present information.

S: A dollar amount.

[9]
P: Give the ic $25000 as the value of old age benefits under employee benefits.

Behavior: Typical imperative request.

S: By under employee benefits,
do you mean to put the particular
roleset and irole or just the irole?

Behavior: Beginning of a clarification exchange.

[10]
P: Can you display just the roleset?

Behavior: Answering a question with a question is typical in the clarification exchanges. Note that "Can you" is really asking about the system's abilities and in this case is not an indirect request for acting.

S: No.

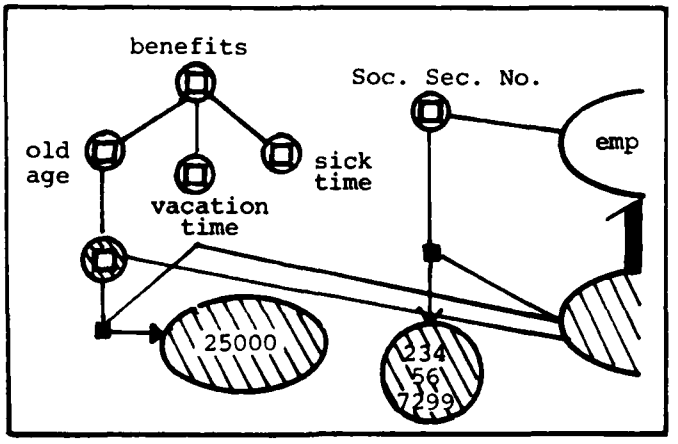
[11]
P: Use the irole. No, never mind.
Display the roleset and irole.

Behavior: Signifying a change of plan by "never mind" is common.

S: Done.

[12]
Now I need some figures on
overhead and R&D.

Behavior: System must recognize that user's last plan is now complete and that his attention is directed to something new. The system is not change the display to overhead and R&D), unless it has some previous knowledge of an interaction that indicates that it knows what the user has in mind by "some figures on overhead and R&D."



This scenario does not exercise all the tasks of KL-ONE-ED described earlier, but for a portion of them, it does demonstrate how people use natural language in a natural way. The scenario illustrates a variety of sentence types and a number of different ways speakers express their intentions. We believe that these forms should be the core of a natural language understanding system for KL-ONE-ED, so that additional tasks may be added without new and extensive changes to it.

4. TOOLS FOR SYNTACTIC AND SEMANTIC INTERPRETATION

In this chapter we report on our progress on the tools we have been building and extending for syntactic and semantic interpretation. We discuss the interaction between the RUS parser and the PSI-KL-ONE parser, and we report on the new dictionary system, which provides lexical acquisition to the RUS parser in a manner that is easy for users to understand.

4.1 Architectures for Syntactic and Semantic Interaction

by R. J. Bobrow and B. L. Webber[2]

One of the major features of the RUS and PSI-KL-ONE components of the Natural Language Understanding (NLU) system is the style of interaction between these two components, including the requirements that such interaction puts on the internal operations of both the syntactic (RUS) and semantic (PSI-KL-ONE) processors. The interaction is characterized by the information flows between two potentially parallel processes, which often occur before either process has completed its task. Thus, before a complete syntactic structure has been assigned by RUS, PSI-KL-ONE is already operating on partial results hypothesized by RUS. Data obtained by Phil Cohen (indicated below) led us to examine the possibility that such an architecture extends throughout the

---

[2] Sections 4.1 through 4.5

entire NLU system. In trying to understand how to design an NLU system with such an architecture, we abstracted and generalized the style of interaction that occurs between RUS and PSI-KL-ONE, and discovered that this style is potentially applicable to a variety of search and perceptual tasks. The following material, based on work by Bobrow and Webber, provides both a general description of this process architecture, and an account of how the RUS/PSI-KL-ONE interaction fits within this framework.

It is generally agreed that determining the syntactic structure of an utterance is only part of a larger process of recovering the meaning, intentions and goals underlying its generation. This implies that parsing strategies cannot be evaluated *in vacuo*. Either explicitly or implicitly, evaluation must take account of the architecture of the NLU system in which parsing takes place. In this chapter we present one such general architecture, together with an evaluation criterion for overall system and component performance. We examine the implication of this architecture for the design of components of the NLU system, particularly the syntactic and semantic analysis processes and their interfaces to one another and to later components such as pragmatics, reference resolution, discourse modeling and response generation. Our interest is in exploring *subarchitectures* for such systems (types of interconnections, the "whens" and "whats" of information flow, etc.) that can maximize system efficiency.[3]

[3] Such subarchitectures are are contrasted with increasing efficiency by recoding, compiling, etc.

This model forms the basis for a family of NLU systems developed at BBN. In this chapter, we both discuss general design issues and contrast the particular implementation decisions made here with those made in other systems.

The general architecture derives from a view of natural language understanding as a perceptual process, that is, as a process which attempts to account for input data in terms of some underlying object or event that produced the data. We assume that utterances are generated for a purpose, and that underlying them are meanings and a range of intentions which a listener must attempt to recover in order to respond appropriately. Natural language understanding involves giving an account of the words of an utterance in terms of syntactic functions, both of these in terms of a propositional content, and all three in terms of a range of speaker meanings, intentions, and goals (from directing the listener's attention to a particular object to getting some nonlinguistic task accomplished). At any instant such accounts should be the best ones the listener can muster, but this by no means implies that these accounts cannot or will not be refined as the discourse proceeds and more data must be fit into place.

Since the process of generating an utterance involves the

application of many sources of knowledge and many constraints[4], it is plausible to assume that understanding - the inverse of the generative process - involves similar types of knowledge. The architecture we are considering assumes that understanding can be modeled as an interconnected set of processors with a primary direction of information flow through them, intent on applying such knowledge.

This is not a new idea, especially in the artificial intelligence community. Early on, it was common to find the following architecture in natural language question/answering systems:

where arrows indicate the primary (in fact, the only) direction of information flow through the system. Even the idea of increasing the richness of connections in such systems - additional feedback and "feedforward" channels - is not a new one. For example, even the earliest version of the LUNAR system [72, 73] had a communication link from its semantic interpreter

---

[4] These constraints come in part from the planner, which chooses both the meaning to be expressed and the focus to be given to meet some set of communicative goals. Further constraints are added by the semantic component, which suggests words to express the meaning and by the syntactic component, which constrains how these words can be joined to produce a well-formed structure that carries both the meaning and the focus.

back to its ATN parser; this link was used to tell the parser that the syntactic structure it had assigned to the input sentence was uninterpretable and hence couldn't lead to a retrieval request -- "Try again." The link provided an information path from the semantic component to the parser, but it required the parser to produce a complete syntactic analysis of the input without semantic feedback to advise its intermediate decisions.[5] terminology, and the interpreter communicated back to the parser whether or not the constituent was semantically acceptable. Unfortunately, this step slowed down the process considerably because the interpreter was being asked to evaluate many more constituents than would be part of the final syntactic structure assigned to the sentence, and the attempt was abandoned.) It also had the effect that no output could be obtained from the system unless and until the parser was able to produce a complete syntactic analysis of the input string.

These characteristics of the information flow in the original LUNAR system depart from the performance of human listeners. Even before an utterance is complete, it appears that listeners begin to generate an explanation for it. There is even evidence that under certain conditions, they begin to respond to

---

[5] Later, a simple experiment was done on LUNAR to see the effects of such advice. The parser informed the interpreter of a constituent about to be closed or popped, in augmented transition network (ATN

it well before it is complete.[6] These interim explanations and responses may later be modified or replaced, but they are correct in a surprising number of cases. In any event, the overall NLU system should not only produce explanations of complete utterances, itself a non-trivial task, but should also lend itself to a style of processing that has been referred to as producing continually available output [41]. Having such continually available output from intermediate processors also permits the design of a system in which the various processors can interact to improve the overall efficiency of the recognition process. The notion of a recognition process with continually available output forms the intuitive basis for what we are calling "incremental *recognition*" - the subarchitecture for NLU systems that we are here advocating.

The idea of using semantic information to aid in the parsing process is highly seductive, and several directions have been taken in an attempt to achieve this goal. "Semantic grammars" [17] finesse the issue by discarding modularity in favor of absorbing parsing and interpretation into a single function. More modular approaches to natural language understanding systems

---

[6] This process has been observed in videotapes of students performing assembly tasks in response to verbal instructions. Phil Cohen reports that in one case the student initiated a response that seemed to require determination of the intended speech act, but that occurred before any syntactic constituent had been completed.

like Diamond/Diagram [44, 54] continue the attempt to use "later" processes to improve the performance characteristics of both "earlier" processes and the system as a whole. So this chapter should not be read as claiming the first or the only exploration of rich interconnections between the processes involved in natural language understanding. Rather, it is a discussion of design criteria we are developing for such interconnected systems - criteria that maximize their efficiency, as well as lend themselves to producing continuously available output. We also discuss the system we are designing and building to meet these criteria.

Since the design criteria we propose are based on the idea of "incremental recognition," we must first explain what we mean by "recognition" and the way in which it can be considered "incremental." This explanation will bring up several important design issues, including:

- The importance of coordinated design of pairs of processes and the information and control flow at their interface
- Strategies for specifying and using partial results - i.e., abstract vs. disjunctive descriptions
- Operating with incomplete input.

Before we engage in more abstract rambling though, we will first introduce the example from which our theoretical notions were derived. We will give a brief overview of the operation of

BBN's RUS[7] parser and the syntactic-semantic interface and interpretation system PSI-KL-ONE[8], whose style of interaction led us to investigate the general concept of incremental recognition processes. Both RUS and PSI-KL-ONE may be viewed as incremental recognition processes of different types, and their cascaded interaction[9] produces another incremental recognition process, which has been used as the front end for several NLU systems. After this overview of RUS and PSI-KL-ONE, we will give an introductory definition of the concept of incremental recognition and the design issues it raises for the construction of NLU systems. Finally we will discuss the possibilities for extending the current RUS/PSI-KL-ONE interaction to permit the design of a NLU system in which it will be possible to generate initial output from the discourse and response planning components even before the parser has completed parsing any constituents.

4.2 The Syntactic/Semantic Cascade

---

[7] An acronym for "Render Unto Syntax."

[8] "Parsing and Semantic Interpretation in KL-ONE"

[9] A generalization of the concept of a pipelined interaction of processes, see [76].

4.2.1 The RUS parser

In Section 4.1 we mentioned an experiment done to see if LUNAR's performance could be improved by making its ATN parser and semantic interpreter more incremental. This experiment involved asking the semantic component to verify the interpretability of each syntactic constituent built, as it was about closed, or popped, in ATN terminology. LUNAR's performance was not improved, and this particular approach was abandoned. However the idea of incremental syntactic/semantic processing was not summarily abandoned. An investigation into the efficiency advantages of "semantic grammars" [17], coupled with a desire to retain the modularity and extensibility of a purely syntactic parser, led to the development of the RUS parser.[10]

RUS was based on the notion that both the semantic interpretation process and the possibilities for feedback from the semantic component to the parser would be better served if there was interaction between the parser and the semantic component at the time that phrases were about to be attached as constituents of larger phrases, as well as at the time that complete phrases were about to be closed, or popped. That is, the semantic component can provide more useful feedback, at a

---

[10] The acronym RUS, for "Render Unto Syntax", was suggested by Dick Burton to indicate that RUS made use of semantic constraints like a semantic grammar, but still gave prominence to the syntactic regularities shared across all semantic domains.

more opportune time, if it is asked whether it is reasonable to attach some newfound constituent as part of the current syntactic phrase. At this point, it can also incrementally add to its developing semantic characterization of the phrase. This allows immediate useful feedforward - supporting the notion of "continuously available output". And moreover, when the phrase is closed (popped), its full semantic characterization is available, from which its interpretation can be derived.

The RUS parser has provided the framework for a number of different NLU systems developed over the last several years at BBN [4, 37]. The RUS parser itself is written as an augmented transition network (ATN) [71] grammar, which is highly efficient because of its novel control structure[11] [4, 5, 6] and its translation into an efficient INTERLISP program by a grammar-compiler [16]. RUS interfaces cleanly with the semantic component, and the identical grammar/parser has been used in all systems, independent of the choice of semantic interface and interpretation mechanisms.

---

[11] The current RUS parser operates without backtracking in a large number of the difficult cases handled by Marcus' Parsifal system [36], due to several variations from normal ATN depth first control structure. Recent work on RUS has involved providing a generalized, symbolic scheduling mechanism, straightforward relaxation of syntactic and semantic constraints for partially ill-formed input, and the ability to use the semantic component to simultaneously evaluate alternative attachments for optional prepositional phrases (LUNAR's "selective modifier placement" facility).

Within the RUS framework, the interaction between the parser and the semantic component takes place incrementally as the parser scans the input string from left to right, one word at a time. A semantic characterization of each syntactic constituent is developed in parallel with determining its syntactic structure, on the basis of constraints on syntactic realizations of particular semantic meanings. Knowledge developed in the course of producing this characterization is fed back to control further action by the parser.

In early RUS-based systems, this characterization was equivalent to the semantic interpretation of the phrase. In the current RUS/PSI-KL-ONE system, it is not. Rather, it is based on the syntactic structure of the phrase and the semantic interpretation of each of its already assigned constituents - what we call its syntactic/semantic shape. It is the job of the syntactic/semantic interface -- one half of the PSI-KL-ONE system -- to develop this characterization, which in turn determines the phrase's susceptibility to various classes of interpretation rules. It is the job of the semantic interpreter -- the other half of PSI-KL-ONE -- to find these rules and compute an interpretation. This job, as we shall show in Section 4.3, is done in an extremely efficient way.

Within the RUS framework the parser and the semantic component engage in a dialogue consisting of transmissions from the parser and responses from the semantic component. Each

transmission represents the communication by syntax of some critical incremental hypothesis or decision made in the parsing process. The information thus communicated is used by the semantic component in characterizing the utterance, and (since it is posed as a hypothesis to be accepted or rejected by the semantic component) provides a means for the semantic component to give feedback on the semantic plausibility of the syntactic structure being proposed.

The most common type of transmission is a proposal by syntax to attach some previously parsed and interpreted constituent phrase to the phrase currently being analyzed. This interaction at the time of constituent attachment appears to be critical to the performance of the overall system. Such attachment transmissions[12] propose that some specific functional relation holds between a previously parsed and interpreted constituent and the matrix phrase whose parsing and interpretation is in progress. The proposal takes the form of a matrix/label/constituent triple. The semantic component either rejects the proposal or accepts it and returns a pointer to a data structure that represents its knowledge of the resulting

---

[12] We use an extended notion of functional relation here that includes surface syntactic relations, logical syntactic (or shallow case structure) relations, and relations useful for determining discourse structures such as primary focus. We talk about "labelling" a constituent's functional role within a matrix. For example, a noun phrase (NP) can serve various functions in a clause, including logical subject (lsubj), logical object (lobj), surface subject (ssubj), and first NP (firstnp).

phrase. (This pointer is not analyzed by the parser, but is rather used in the description of the matrix that syntax includes in its next proposal (transmission) to extend the matrix.) As noted earlier, the parser is implemented as an ATN, and transmissions occur as actions on the arcs of the ATN grammar. The failure of an arc because of a semantic rejection of a transmission is treated exactly like the failure of an arc because of a syntactic mismatch; alternative arcs on the source state are attempted, and if none are successful, a backup occurs.

4.2.2 PSI-KL-ONE

In this cascaded system, the semantic component has two related tasks to perform:

1. provide feedback to the parser by checking the semantic plausibility of the proposed functional labels for hypothesized constituents of a phrase, and
2. build semantic interpretations for individual phrases to feed forward to the rest of the system.

In PSI-KL-ONE, these tasks are performed separately by two modules which are themselves cascaded - the syntactic/semantic interface and the semantic interpreter. Both make use of the ability of our representational formalism KL-ONE [12, 11] to represent a virtual, lattice-structured taxonomy of patterns and to provide general inheritance mechanisms within this taxonomy. The taxonomy is used to characterize efficiently both the syntactic/semantic shapes of interpretable phrases (what in LUNAR [72] would correspond to the patterns of its semantic

interpretation rules) and as allowable ways to rewrite general or partial shape descriptions into more specific ones. The inheritance mechanism is used to store the component actions involved in the construction of a phrase's interpretation from the interpretations of its constituents (what in LUNAR would correspond to the action parts of its semantic interpretation rules).

We introduced the concept of a syntactic/semantic shape specification in terms of the semantic interpretation of the constituents of a phrase and the syntactically determinable functional relations by which these constituents were attached. For the purposes of building semantic interpretations, as well as providing feedback to the parser, we are interested in a deeper or more semantic notion of shape. In such a deep syntactic/semantic shape description, the relation between a constituent and the matrix to which it attaches belongs to a set of extended case frame or semantic relations. A semantic relation (or semantic role) completely specifies the way in which the interpretation of the constituent is used in constructing the interpretation of the matrix phrase.

The feedback provided to the parser from the syntactic/semantic interface is based on the assumption that every constituent of a phrase serves some semantic or discourse-level purpose. In particular, each constituent must contribute

to the meaning of the phrase[13] by filling some particular semantic role. Of course, the mapping of functional labels to semantic relations is clearly not one-to-one, nor is it unconstrained. For example, the logical subject of a clause whose main verb is "hit" might be the agent of the act (e.g., "The boy hit ...") or the instrument (e.g., "The brick hit ..."). There are constraints on which functional labels can be mapped to which semantic roles, and these constraints also depend on the semantic interpretation of the constituents involved. Thus, checking the semantic plausibility of a syntactic attachment proposal involves determining if there is an acceptable mapping between a proposed functional label and one or more semantic relations. If such a mapping is not possible, then the proposed attachment is not semantically plausible.

The PSI-KL-ONE system uses a set of rewrite rules (which we call relation mapping rules or RMRULEs) to specify what mappings are possible between functional labels and semantic relations. These rules, which are stored in the taxonomy, indicate the conditions under which a given functional label can be mapped into a semantic relation. They thus determine how (if at all) a "syntactically specified" syntactic/semantic shape description

---

[13] For the moment we lump discourse effects with the semantic component, so that we include constituents that have no obvious semantic purpose, but act only as discourse markers. These can readily be included in the overall strategy.

can be mapped into a more "semantic" one. The pattern of an RMRULE specifies the conditions under which it matches a syntactic proposal, in terms of:

- the syntactic shape of the matrix (e.g., "It is a transitive clause whose main verb is 'run'."), and the interpretation and semantic role assigned to other constituents (e.g., the constituent labeled lsubj must be interpretable as a person, and must be the agent of the clause"),
- the proposed functional label, and
- the interpretation of the constituent to be added.

If the pattern matches, the proposed functional label connecting matrix and constituent is replaced with an appropriate semantic relation. (The syntactic/semantic shape of the matrix and the interpretation of the constituent may be refined in the process as well. See Section 4.3.)

Associated with each semantic relation connecting a constituent and matrix description are one or more component actions that specify how the interpretation of the constituent is to be used in building the interpretation of the matrix phrase.[14] After all the constituents of a matrix have been found and assigned appropriate semantic relations, the interpretation of a phrase is built up by executing all of the actions that apply to

---

[14] In our AAAI paper [6] we referred to these rules as "interpretation rules" or "IRULEs," but in this chapter we reserve those terms for a higher level abstraction that combines some of the features of both RMRULEs and actions. We discuss such IRULEs in section 4.3.

the phrase. This buys efficiency by rejecting attachments which have no hope of semantic interpretation, while deferring the construction of an interpretation until the syntactic well-formedness of the entire phrase is verified.

We shall give a more detailed example of the operation of the RUS/PSI-KL-ONE syntactic/semantic cascade in Section 4.3.

4.3 Incremental Recognition

4.3.1 Informal definitions

In order to characterize the operation of the RUS/PSI-KL-ONE system in a way that can be used to highlight some general issues in the design of NLU systems, we introduce the concept of a system built as a cascade of incremental recognition processes. Informally, recognition involves explaining a set of data in terms of some source (an object or process) which could account for or generate it. Since an explanation is itself a description (i.e., one is not actually handing over or pointing to the source object or process), the closest one can come to providing such an explanation is by describing the source of the data. Thus, at some level recognition involves accounting for one description or

data set in terms of another.[15]

More precisely, recognition is a process that maps an *input description*, consisting of a set of data expressed in some *data language*, into an output expressed in some *target language*.[16] To specify a recognition task, one must (at a minimum) give both the data and target languages, as well as a *compatibility relation* between input data sets and output descriptions. In general, for each input data set there is a non-empty set of *compatible descriptions* in the target language. The simplest recognition task merely involves finding one such compatible target description given a data set. A harder task is to characterize, either extensionally or intensionally, the entire set of such compatible target descriptions.

Most often however, the goal is to find "the best" such compatible target description. Intuitively, this description

---

[15] For recognition to be worthwhile, the output description must have some characteristics that make it a reasonable goal. Where there is some generally accepted theory or model of the underlying structure of data sources, that theory provides a useful target. In general, the output description must merely be, in some sense, a more *useful* representation of the underlying source than the data to be accounted for. The utility of the output representation depends heavily on the processes that take it as input. In general, a representation is more useful if it makes it easier to compute and represent other knowledge, such as constraints and necessary inferences.

[16] Both the input language and target language define sets of expressions that may be infinite. Moreover, the target language for one recognition process may be the data language for another.

corresponds to the source "most likely to have generated the given data." While the compatibility relation is posed as being independent of the context in which the recognition task is performed, the likelihood of various sources usually depends on the context in which the data are obtained. To account for context, we assume the existence of a separate evaluation process that takes an input description and a compatible target description and returns a number (say between 0 and 1) corresponding to the contextually determined goodness of their match.

Thus, for any particular input description, recognition involves finding a target description that is compatible with the input and which matches the input better (in terms of the evaluation process) than any other compatible target description. In general, this quality gives recognition processes some flavor of a "search," although the search may appear in many guises, such as sequentially generating and comparing the goodness of individual descriptions, operating on entire classes of descriptions seeking to constrain the form of the optimal description, or actually following some path through the space of descriptions under the direction of a general search algorithm.[17]

---

[17] Note that some problems can be approached as recognition problems or, alternatively, analysis problems. A good example is that of fitting a line to a set of points. This can be viewed as searching for the right line among the space of possible lines or, alternatively, computing the coefficients for the line mathematically via regression/Gaussian analysis. Thus not all recognition problems need be solved with recognition processes.

The characterization of what it means for a process to be, in some useful sense, "incremental", depends strongly on the architecture of the system in which the process is embedded. As noted earlier, our primary concern is system architectures in which the optimal target description produced as output by one recognition process is used as the input to a later process. We are also concerned with a slight variant of this architecture in which the evaluation process described above is performed not locally, but by a later process. This variant provides a type of feedback loop in the overall system. Such feedback allows later processes to control the operation of earlier ones, perhaps redirecting their search on the basis of knowledge unavailable at an earlier stage. To implement such feedback, more than one compatible target description may be passed along to later processes, either simultaneously or sequentially, as alternative hypotheses. Later processes must treat such "hypothesized" descriptions differently than target language descriptions that are "certified optimal."[18] We shall have more to say about this architecture in Section 4.3.2.

In both architectures, there are advantages when an earlier process makes available certain types of partial results as soon

---

[18] This is a plausible way to view some of the interaction between a syntactic processor and semantic and discourse processors: a syntactic processor can only suggest a range of possible attachments for such modifiers as optional prepositional phrases, while it is up to the semantic processors to determine the best placement.

as possible - later processes can then generate and feed forward output in a pipelined fashion and can also give feedback to the initial process, guiding its generation of target descriptions.

The ability to make intermediate results continually available to both later and earlier processes is the fundamental characteristic of an incremental process, and the ability of those other processes to make good and timely use of such information is the measure of the degree of "useful incrementality" of the overall system. Therefore, any design decisions for useful incrementality must involve pairs of processes. Before we attempt to give a more rigorous characterization of this notion of incremental computation, we will continue at an intuitive level within the NLU domain.

4.3.2 Informal examples

RUS/PSI-KL-ONE provide some simple examples of "incremental recognition". To begin, consider a sentence starting "John ran ....". While there are many types of "run" sentences, both transitive and intransitive, with a variety of different meanings - no matter how the above sentence continues, any structural description output will have the feature "subject = John."[19]

---

[19] This conclusion cannot be reached after just the first word "John" has been processed: some extensions compatible with that will have the feature "subject = John, others "object = John," etc. Before this feature can be known for certain, at least the entire auxiliary structure of the sentence must have been processed.

What advantages follow from feeding this information immediately to other processes?

Recall PSI-KL-ONE'S RMRULEs described earlier, which, by specifying the syntactic/semantic shapes of interpretable patterns, essentially represent constraints on the syntactic ways to express certain meanings. They provide two ways in which information from the parser can be made useful. First, their results can be fed back to guide the growth of the syntactic tree, and second, they can allow the semantic interpreter to get started. For example, with a "run" sentence, if the parser can identify its subject incrementally, as above and pass that information on to the semantic component, rules may be found for constraining the sense of "run" to either transitive (e.g., "John ran the race," "John ran the computer") or intransitive (e.g., "The program ran for 50 seconds on a PDP-11/60"). These rules may then be used to generate feedback to the parser, so that, for example, if the intransitive sense of "run" applies, the parser does not need to consider arcs leading to a direct object. As for semantic interpretation, knowing that the subject of "run" was John guarantees an agentive sense of the verb (with John as agent), even before one learns the syntactic identification and semantic characterization of its object.

Notice that this interaction depends on both the parser and the semantic interface operating incrementally. On the one hand, the parser must feed forward information on the logical subject

and main verb of the sentence as soon as they can be guaranteed on the basis of syntactic evidence. On the other, the semantic interface must immediately determine that in the "program run" case, for example, the only applicable RMRULEs are ones that require the main verb "run" to be treated in its intransitive sense. If the semantic component did not check patterns for RMRULEs until all potentially relevant information was available, it would not be able to feed back constraints to the parser.

Note that we can view the job of the syntactic/semantic interface as continually refining and restricting the set of patterns and the set of actions that might possibly be applicable to the matrix phrase under construction. Its ability to provide feedback and feedforward depends on its ability to recognize constraints shared by all remaining rules. It can give feedforward whenever the set of applicable actions implies that some property is true of whatever the resulting interpretation is, assuming that the phrase is semantically coherent. It can give feedback whenever the remaining set of patterns (RMRULEs) is empty or resquire that some constraints on future attachments hold, no matter which rules are finally applied. Notice that as more is learned about the phrase, the set of possible patterns and actions may contract or stay the same, but it never grows.

4.3.3 Local vs. non-local evaluation

We now proceed to give a more general characterization of "incremental recognition." Consider a process which takes as

input a set of data in some data language (e.g., a string of words in some natural language) and produces an output in some target language (e.g., a parse tree for some syntactic derivation of the word string). We further assume that the input is processed sequentially by the recognition device. Thus, at any time, the processor has processed some initial sequence, say d(1)...d(k). Now suppose that after some such initial sequence, every compatible extension[20] had some feature f in the target language[21]. While f might be further refined as a result of acquiring more data, any actions that later processes can take on the basis of f will be consistent with whatever the optimal target language description turns out to be. This is the situation we discussed earlier, where we noted the benefits of making such shared, albeit partial information available.

---

[20] We assume for generality that not all sequences of input data are "valid" -- i.e., not all sequences can be mapped onto a target language output. (Such sequences might correspond to syntactically ill-formed word strings). Thus, there are some sequences for which the device can assign a target language description (e.g., where d(1)...d(k) constitutes a complete sentence), others for which any such description is ruled out (e.g., a word string for which there is no syntactically acceptable extension), and still others that haven't yet provided enough information to pin down a description or rule them all out (e.g., a word string that is the initial segment of one or more sentences). In the last case, we term "compatible extensions" those final sequences of data $d_{k+1}...d_n$ that allow the recognition device to assign a target language description to the whole.

[21] E.g., all syntactically well-formed continuations of "The boy sees..." have the NP "The boy" as **firstnp**, **ssubj**, and **lsubj**.

There is an extension to this notion that captures an important aspect of the interaction of RUS and PSI-KL-ONE. Previously, we described an important variation on the above architecture in which an earlier process can generate only the set of compatible target language descriptions for an input. In such cases, a later process is responsible for evaluation of the descriptions in the set and selection of the optimal one. In this architecture, there is often no feature f common to all compatible descriptions. Thus no single feature can be "guaranteed correct" and passed along for appropriate action by later processes. However, the principle of continually available output can still be applied to this architecture for recognition as search.

Consider a simple case of later evaluation and selection, where evaluation of a target language description is limited to a 0/1 decision (i.e., impossible/possible), rather than spanning the range between them. Given just an initial sequence of data there may be no way for the earlier recognition process to determine on the basis of the data thus far which of several alternative target language features to assign to it. In such a case, if another process can make an acceptance/rejection decision locally and cheaply, then it may be worthwhile for the first process to hypothesize a compatible target language description and pass it on for evaluation. Using the resulting feedback, the first process may come to an optimal target language description faster than it could do so on its own, if indeed it could come to such a description at all.

For example, in an active "send" sentence like "Smith sent a message...," the parser cannot immediately determine whether the first NP after send is the direct or indirect object. Only after determining whether there is a "to"-PP or another NP directly following can that decision be made. However, the semantic component may well be able to rule out one or the other role immediately. To take advantage of this semantics capability, the parser could propose to the semantic component that the NP "a message" is the indirect object of send. The semantic component could then reject this proposal on the basis of what it implies semantically - there is no sense (in the domain being modeled) in which a message can be sent anything.

This use of other processes to make decisions is not without its problems. For example, RUS will currently transmit to the syntactic/semantic interface a proposal to attach a newly constructed prepositional phrase to a particular open matrix. If the proposal is rejected, RUS must merely come up with another syntactically compatible description. However if it is accepted (i.e., local evaluation of the hypothesis is 1), a problem may arise. This feature - here, a particular attachment - may not be shared by all compatible target language descriptions: there may be alternative values for the feature (i.e., alternative attachments) whose local evaluation is also 1. In this case, the later process cannot treat the hypothesized feature in the same way it would treat a necessarily shared feature, since it may later be found incompatible with the remaining data. Currently

RUS/PSI-KL-ONE records enough information to allow backup over hypotheses later found to be incorrect, but it takes advantage of the fact that the output of the semantic interpreter is not fed on incrementally to other processes, and hence not much has been risked. Before closing this section on local vs nonlocal evaluation, we want to make two further distinctions. The first concerns ambiguity, and the second, the nature of the description sent on to other processes. As to the first, in many cases an initial subsequence of data appears ambiguous because the recognition device has not yet received enough information itself to decide among the alternatives. Thus in the "Smith sent the message" example above, the parser could wait until after it had verified the nonexistence of a second NP immediately following "the message" to conclude, without semantic feedback, that the first NP was the direct object.

However, in many other cases, it is clear that no future input will resolve the problem: as far as the given recognizer is concerned, there is true ambiguity. For example, as Marcus [36] has noted, in cases combining both dative movement and WH-movement of a verb's direct or indirect object (e.g., "Which boy did John show the cat?," "Which fish did John feed the boy?"), syntactic evidence alone is not sufficient to make correct direct and indirect object assignments. The attachment of optional post-modifiers, such as prepositional phrases and relative clauses is also a case of true ambiguity for syntax, as is the determination of quantifier scope [5, 66] a case of true

ambiguity for semantics. In these cases, it is not simply a matter of efficiency whether a set of alternatives, or a hypothesized feature value, is transmitted to other processors -- there is never a point at which the ambiguity is resolvable within the processor nominally responsible for providing the class of feature involved.

Finally, we want to distinguish the nature of the alternative compatible target language descriptions that can be sent forward. Such a description can take the form of either a disjunction (with the possibly high cost of specifying all the disjuncts) or an abstraction. The receiving process is left with the problem of refining the choice to get the optimum, or in the 0/1 case, the alternatives evaluating to 1. For example, in the case of PP attachment, RUS could send a disjunction of specific attachment proposals, or it could send a single "abstract" proposal annotated with the possible attachment points.

In the design of PSI-KL-ONE, we have endeavored, wherever possible, to design representations and processes that worked with <u>abstract</u> rather than <u>disjunctive</u> representations. Why is this the case? As already noted, there is the possibly high cost of specifying the disjuncts - (1) because there may be many of them, and (2) because each disjunct is a self-contained unit. On the other hand, one way of viewing an abstract representation is that it specifies just those <u>relevant</u>[22] features that are shared

[22] That is, useful for future processing.

by all the alternatives represented. Features that are not shared or relevant are suppressed, so that the representation of the set actually takes less information than the representation of any single alternative. Such a representation is only possible where the desired set can be specified as just those alternatives satisfying the common properties.

There are also processing advantages to working with abstract rather than disjunctive descriptions. In general these advantages have to do with avoiding the case analysis necessary for processing straightforward disjunctive descriptions. With such descriptions it is necessary for processes to consider each disjunct individually and to derive the results of each. Certain strategies can cut down the amount of repeated work, but these strategies (such as "memo-izing" common results, as in a parser's "well-formed-substring table") all exact a certain penalty in overhead. Moreover, some effort may be required to discover that the ways in which disjuncts differ are irrelevant to some computation. There is an advantage if the original process has knowledge of the way in which features of its outputs will be used, and can produce representations that factor out useful shared properties for future consideration. In the next section we discuss a particular incremental recognition process used in PSI-KL-ONE's syntactic/semantic interface; this process is based on an explicit representation of appropriate abstractions within the target language. It belongs to a class we refer to as the incremental description refinement processes.

4.4 Incremental Description Refinement

We noted in Section 4.2.1 that the job of PSI-KL-ONE's syntactic/semantic interface was to develop a semantic characterization of each phrase, in parallel with the determination of its syntactic structure. That semantic characterization is not the semantic interpretation of the phrase, but rather a description that mediates between the output of the parser and the input of the semantic interpreter. That description is based on a combination of the syntactic structure of the phrase and the interpretations of its constituents,[23] what we have called its syntactic/semantic shape. In this section, we discuss how the interface performs its task using an incremental recognition process we call incremental description refinement, or IDR.

An obvious first question is why have such a mediating description? The reason is to improve the efficiency of the semantic interpreter. We noted in Sec. 4.2.2 the relationship between LUNAR semantic interpretation rules and PSI-KL-ONE's taxonomy of syntactic/semantic shapes. In LUNAR, the interpretation of a phrase or utterance depends on the interpretation rules whose patterns it matches. The actions of these rules are executed in some order, to produce a structure

---

[23] They have already been produced by previous interaction between RUS and PSI-KL-ONE.

representing the interpretation of the phrase.[24] However, a major portion of the LUNAR interpreter's work is finding the set of rules whose patterns match the given phrase. This process is improved by identifying the phrase's syntactic/semantic shape (and also facilitates feedback to the parser, as noted in Sec. 4.3.2). How is it done?

Viewed as a recognition device, the data language of the syntactic/semantic interface is (roughly speaking) the set of syntactic functional descriptions of phrases producible by the RUS parser,[25] while the target language consists of syntactic/semantic shape descriptions. These are represented as Individual Concepts within the virtual lattice-structured taxonomy of KL-ONE Generic Concepts that represent more or less specific (and/or complete) interpretable syntactic/semantic shape descriptions. These Generic Concepts correspond to the patterns of LUNAR's interpretation rules. Efficiency derives from the taxonomy's making it possible to search <u>in parallel</u> for all interpretation rule patterns that are compatible with a partially parsed phrase (and hence, all appropriate IRULEs as well). The

---

24 For example, in LUNAR the actions of its interpretation rules often recursively invoked the interpretation process on constituents of the current phrase in a top down fashion.

25 It is essential to note, however, that these are not purely syntactic descriptions -- as a result of recursive interaction between RUS and PSI-KL-ONE, every constituent of the phrase under consideration by PSI-KL-ONE has already been interpreted.

only problem is that such searches must be made in conjunction with the other task of the interface, that is, providing feedback to the parser about the semantic plausibility of such syntactic decisions as constituent attachment and functional labeling. That task requires the interface to keep track of the set of all syntactic/semantic shape descriptions which might potentially be applicable to a partially parsed phrase. In general, each new hypothesized syntactic structure will reduce the size of the set of compatible descriptions. Semantic rejection is signalled in just those cases when the set of compatible descriptions becomes empty.

For example, the pattern informally described as

- a clause whose head[26] is the verb "hit", whose logical subject is interpretable as a movable object, and whose logical object is interpretable as a physical object

might be represented by the KL-ONE structure in Figure 9.

There is normally a great deal of redundancy in the set of patterns that forms the left-hand-side of the interpretation rules embodied in any semantic interpreter of reasonable size. The structured inheritance mechanism provided by KL-ONE allows us to abstract the common portions of several patterns and to represent these shared subpatterns as high-level, or abstract,

---

[26] We refer to the main verb of a clause as its head.

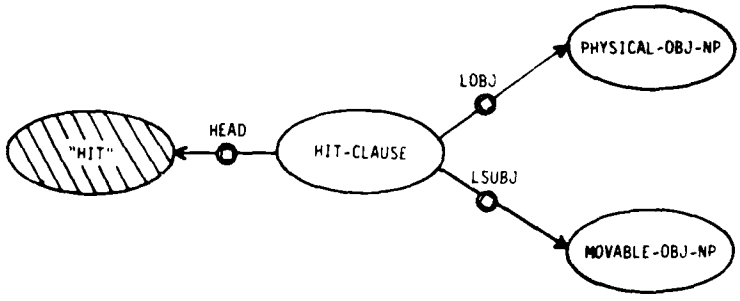


FIG. 9. A SYNTACTIC/SEMANTIC PATTERN REPRESENTED IN KL-ONE.

concepts within the taxonomy of syntactic/semantic shapes. The structure of these abstract pattern concepts is inherited by all concepts representing more specific patterns.

Thus, for example, the two patterns:

- a clause whose head is the verb "hit" and whose logical subject is interpretable as a movable object
- a clause whose head is the verb "hit" and whose logical subject is interpretable as an animate being

share the common part

- "a clause whose head is the verb "hit."

In KL-ONE we would represent this shared pattern by a common super-concept, as in Figure 10[27].

---

[27] To reduce clutter, we have left out several SuperC cables to the Concept NP.

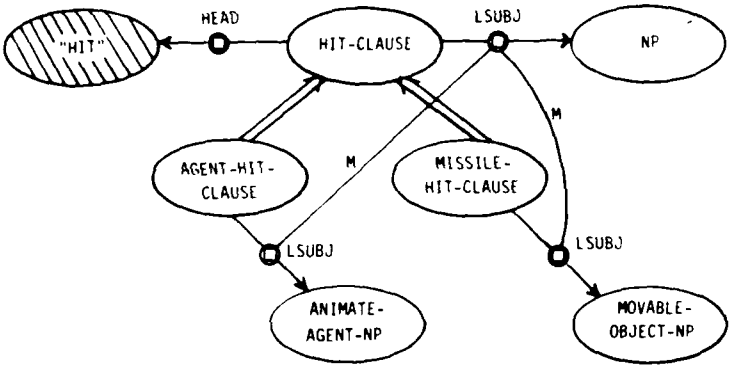


FIG. 10. PATTERN SHARING REPRESENTED IN A PIECE OF A KL-ONE TAXONOMY.

The set of pattern-action rules making up the underlying knowledge basis of a semantic interpreter has even more shared structure. Many rules share common actions. These actions are expressed as ways of mapping from the part of the structure matched by the pattern of the rule to components of the result produced by the rule. Because a finite set of interpretation rules (such as LUNAR's or PSI-KL-ONE's) must apply to any of a potentially infinite set of interpretable phrases, the pattern in each rule must match a large class of phrases. This match is commonly accomplished by having variable elements in the pattern, each of which stands for some portion of the phrase that satisfies a given predicate. Often this element is a syntactic constituent of the phrase, or a constituent of some

constituent.[28] The predicate generally consists of the functional label by which the constituent is attached, as well as some characterization of the constituent's semantic interpretation. The action portion of each rule generally mentions one or more of the variables in the pattern portion, and it uses the semantic interpretation of the constituent bound to the variable in the constructing of the interpretation of the phrase as a whole.

Rules whose patterns share common substructures, often share the way in which the variables in the common substructure are mapped into components of the resulting interpretation. This sharing is codified in the notion of semantic relations (which correspond to pattern variables) and PRULEs in PSI-KL-ONE.

This process of keeping track of the set of syntactic/semantic shape descriptions compatible with a partially described phrase is an example of a more general class of incremental recognition processes, which we refer to as the Incremental Description Refinement (IDR) processes. Informally, IDR is a process of

- o determining the set of target language descriptions compatible with an object or event known to have a given set of properties, and
- o refining that set of descriptions as more properties are learned.

---

[28] LUNAR tagged constituents of interest with integers, so that one talked of "match 1", "match 2", etc.

The significant feature of IDR is that these compatible description sets are represented intensionally. Viewed as intentional representations, at least three well-known methods - decision trees, constraint propagation and version spaces - are types of IDR process. In this section, we show how these methods are IDR processed, and then describe the taxonomic lattice-based IDR process that forms the basis for the syntactic/semantic interface in PSI-KL-ONE.

In an IDR process, one is always working with the complete set of descriptions compatible with the currently known properties of an object - what we'll call its descriptive cover or covering set. As one learns more about an object, its covering set must either shrink or stay the same. Hence the basic steps of any IDR process involve:

1. starting with a set of properties P and the covering set C(P) compatible with those properties
2. when P is extended into a larger set P', computing C(P') by removing inapplicable elements from C(P).

The difficulty is that it is usually impractical, if not impossible, to represent C(P) extensionally: in many cases C(P) will be infinite. (For example, if one considers parse trees as target language descriptions of word strings, until the length of the string is known, the number of parse trees in C(P) remains infinite (or 0), no matter how many words in the string are already known.) Thus, a covering set used in an IDR process must be represented intensionally, with the consequence that "removing

elements" becomes an inference process that determines the intensional representation of C(P') given the intensional representation of C(P). Note that just as any element of C(P), represented extensionally, may be structured, the intensional representation of C(P) may be structured as well.

The trick in designing an efficient and effective IDR process is to choose a synergistic inference process/intensional representation pair. One example is the use of a discrimination tree. In such a tree, each terminal node represents an individual description, and each nonterminal node represents the set of descriptions corresponding to the terminals below it. Every branch indicates a test or discrimination based on some property or properties of the object to be described. Each newly learned property of an object allows the IDR process to take a single step down the tree, as long as the properties are learned in an order compatible with the tree's structure. Each step thus reduces the set of descriptions subsumed.

Another IDR process is the operation of a constraint propagation system [35, 64, 67]. In such a system a known object can be specified in terms of its relevant features, their values, and any dependencies (constraints) among the values assumable by particular subsets of features. Graphically, any such object specification can be represented as a set of nodes corresponding to different features, each labeled with its corresponding value. Nodes/features are linked together with those on which they

depend. Essentially then, a constraint relation specifies which pairs of labels can occur ,on adjoining or, linked, nodes.

When the system is presented with a new and unknown object, it begins by creating a structure that has a set of labels for each feature. As it learns things about the object, it pares down certain feature sets, which by the a priori constraints may cause others to be pared down in turn. At any point, a descriptive cover is simply the cross-product of the node label sets. The refinement operation consists of:

1. extending the analysis to a new node to learn more about it
2. eliminating now incompatible labels
3. removing all incompatible labels from adjacent nodes
4. propagating the effects.

Unlike the use of a discrimination net, constraint propagation does not require that information about nodes be considered in some a priori fixed order.

A third type of IDR process is Mitchell's "version space" approach to concept learning [39]. Concept learning involves forming a general description of a class of objects given a set of examples and nonexamples. In the version space approach, the covering set of all and only those general descriptions which account for the distribution of positive and negative examples thus far seen by the system is represented by

1. a set S of maximally specific concept descriptions - i.e., ones more specific would exclude the observed positive instances
2. a set G of maximally general concept descriptions - i.e., ones more general would include the observed negative instances.

As more training instances are presented to the system, S will either expand or stay the same, and G will either shrink or stay the same, until S and G contain one and the same concept description and the concept is learned.

Mitchell also notes the value of being able to use incompletely learned concepts - i.e., intermediate results. Since it is rare for the available training instances to describe precisely the target concept, it is useful to be able to classify at least some new (nontraining) instances with certainty, without having a precisely specified target. This, his S and G sets provide. That is, if the new instance matches every element in S, it can be classified with certainty as a positive instance of the target, and conversely, if it matches no element of G, it is with certainty a negative instance. Concept learning is almost the mirror image of the recognition processes that are our major interest, but as the above comparison shows, the IDR notion is appropriate to this area as well.

In PSI-KL-ONE's syntactic/semantic interface, the set of properties that one can learn about an object consists of descriptions of constituents and their functional relations to their matrix. Unfortunately, surface variations such as passive

forms, wh-movements, and others make it difficult to assume any particular order of discovery of such properties as the parser considers words from left to right. The recognition task in this case is one in which

- o the order in which information will be presented to the system is not determined beforehand
- o the objects to be recognized are not only structured, but of varied structure.

The first property makes discrimination nets an inadequate recognition method, if we are going to hold to providing continually available output. And the second property makes constraint propagation inadequate, since we cannot postulate a priori a fixed structure to be recognized, whose multiple alternative node labelings must be reduced to a single consistent set. Rather we need to be able to "grow" the appropriate structure and reduce the multiplicity of its labelings as the recognition proceeds.

To meet these constraints we have developed an IDR process called TAXLADR[29] which forms the basis of the syntactic/semantic interface in PSI-KL-ONE. Simply put, TAXLADR uses KL-ONE's taxonomic lattice structure as a generalization of a discrimination tree that is order-independent. The actual operation used in TAXLADR also involves an extended notion of

---

[29] For "Taxonomic Lattice Description Refinement".

constraint propagation operating on nodes of the lattice, and thus the system has interesting analogies to both simpler forms of IDR process.

We introduce the description of the activity of the syntactic/semantic interface IDR by giving a simplified example of the interaction of RUS and the interface in the next section. We then indicate some of the difficulties with the straightforward approach taken in the example and discuss some of the features needed in the actual IDR algorithm.

4.4.1 An example of the cascade

As a simplified example of the parser-interpreter interaction and the use of the KL-ONE taxonomy of syntactic/semantic shapes in this interaction, we will briefly describe the process of parsing the clause "John ran the drill press." The simplified ATN grammar we use for this example is shown in Figure 11.

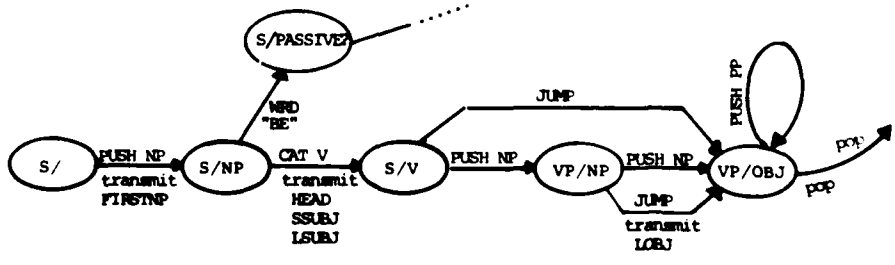


FIG. 11. A SIMPLIFIED ATN.

The simplified taxonomy for our example is given in Figure 12.[30] In the taxonomy, any CLAUSE whose **head** is the verb "run" (independent of tense and person/number agreement) is an example of a RUN-CLAUSE. There are two classes of RUN-CLAUSEs represented in the taxonomy - those whose **lsubj** is a person (the PERSON-RUN-CLAUSEs), and those whose **lsubj** is a machine (the MACHINE-RUN-CLAUSEs). The class of PERSON-RUN-CLAUSEs is again subdivided, and its subclasses are RUN-MACHINE-CLAUSE (in which the **lobj** must be a machine), RUN-RACECLAUSE (in which the **lobj** is a race), and SIMPLE-RUN-CLAUSE (which has no **lobj**).

If we get an active sentence like "John ran the drill press," the first stage in the parsing is to "PUSH" for an NP from the CLAUSE network. For simplicity we assume that the result of this is to parse the noun phrase "John" and produce a pointer to NP1, an Individual Concept which is an instance of the Generic pattern PERSON-NP. This is the result of interaction of the parser and interpreter at a lower level of the ATN.

Since it is not yet clear what role NP1 plays in the clause (because the clause may be active or passive), the parser must hold onto NP1 until it has analyzed the verb. Thus the first transmission from the parser to the interpreter at this level is the proposal that "run" (the root of "ran") is the **head** of a CLAUSE. The interpreter accepts this and returns a pointer to a new Individual Concept CL1, which it places as an instance of

[30] To reduce clutter, we have left out several superC cables to the Concept NP.

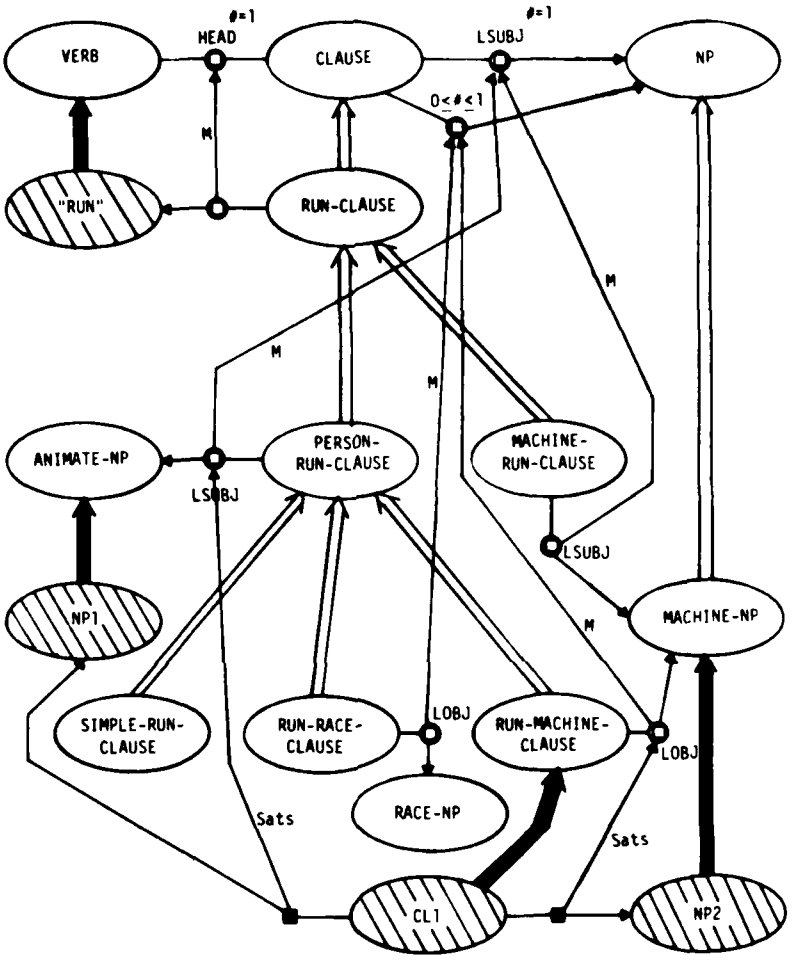


FIG. 12. A SIMPLE KL-ONE SYNTACTIC TAXONOMY.

RUN-CLAUSE.[31]

Since the parser has by now determined that the clause is a simple active clause, it can now transmit the proposal that NP1 is the lsubj of CL1. Because NP1 is an instance of a PERSON-NP, the interpreter can tell that it satisfies the restrictions on the lsubj of one of the specializations of RUN-CLAUSE, and thus it is a semantically plausible assignment. The interpreter fills in the lsubj Role of CL1 with NP1 and connects CL1 to PERSON-RUN-CLAUSE, since that is the only subConcept of RUN-CLAUSE which can have a PERSON-NP as its lsubj.

Finally, the parser PUSHes for an NP, resulting in a pointer to NP2, an instance of MACHINE-NP. This is transmitted to the interpreter as the lobj of CL1. Since CL1 is a PERSON-RUN-CLAUSE, the taxonomy indicates that it can be either an instance of a RUN-RACE-CLAUSE or a RUN-MACHINE-CLAUSE, or a SIMPLE-RUN-CLAUSE. Since NP2 has been classified as an instance of MACHINE-NP, it is not compatible with being the lobj of a RUN-RACE-CLAUSE (whose lobj must be interpretable as a race). On the other hand NP2 is compatible with the restriction on the filler of the lobj Role of RUN-MACHINE-CLAUSE.

---

[31] Actually, the interpreter creates a Generic subConcept of RUN-CLAUSE, in order to facilitate sharing of information between alternative paths in the parse, but we will ignore this detail in the remainder of the example.

We assume that the taxonomy indicates all the acceptable subcategories of PERSON-RUN-CLAUSE. Thus it is only semantically plausible for NP2 to fill the lobj Role of CL1 if CL1 is an instance of RUN-MACHINE-CLAUSE. This being the case, the interpreter can join CL1 to RUN-MACHINE-CLAUSE and fill its lobj Role with NP2, creating a new version of CL1 which it returns to the parser.

At this point, since there are no more words in the string, the parser transmits a special message to the interpreter, indicating that there are no more constituents to be added to CL1. The interpreter responds by finding the IRULEs inherited by CL1 from RUN-MACHINE-CLAUSE, PERSON-RUN-CLAUSE, etc., and using the actions on those IRULEs to create the interpretation of CL1. It associates that interpretation with CL1 and returns a pointer to CL1, now a fully parsed and interpreted clause, to the parser.

4.5 Extensions to RUS and PSI-KL-ONE

In the version of RUS and PSI-KL-ONE currently in operation, the feedback given the parser by the semantic component can only cause the parser to continue or reject a path on the basis of semantic incoherence. However, RUS now has a generalized scheduling structure, which should allow more informative feedback. At the simplest level, semantic processes might extend the notion of acceptance into two cases -- "You're all right - keep going", or "I've now filled in everything I think I could know about one of these - make popping a preference").

4.5.1 Best-first strategies

As we noted in Sec. 4.4, the goal of a recognition process - in our case, natural language understanding - is to come up with the best possible description of the input data. Clearly, this is not what RUS/PSI-KL-ONE currently does: its output is the first semantic interpretation found compatible with the input data, not necessarily the best one. The only procedure we know of that formally guarantees that the first target language description found for the data will be the best one according to some evaluation metric is Woods' Shortfall Algorithm Woods82.[32] This algorithm depends on being able to (1) approximate the best possible score achievable by a complete description and (2) assign to a partial description the best possible score achievable by any compatible extension. Whether it is possible to find the kind of overall "goodness metric" for RUS/PSI-KL-ONE that the algorithm requires demands further study.

4.5.2 Comparative evaluation

Recall that in the current control strategy, the parser explores a single path at a time, only transmitting to the semantic component individual proposals that are consonant with that theory. An alternative strategy more in line with the

---

[32] The standard admissible search algorithm only guarantees that the first path found to a goal will have the lowest cost, again according to some evaluation metric.

requirements of a best-first algorithm involves considering several paths in parallel, transmitting to the semantic component a set of proposals. This strategy is now possible using RUS's generalized scheduling structure. Given a continuous evaluation function, the semantic component might be able to rank order that set by how good the various (partial) matrix interpretations continue to be. (Whether such local goodness implies continued global goodness is a matter for investigation.) But even lacking a continuous evaluation metric - using only accept/reject decisions - the resulting partition of the set of possibilities might provide useful direction to the parser about the best way to continue.

One place where it is important to be able to compare alternatives is in the attachment of modifiers, such as prepositional phrases. Here we can take advantage of RUS's scheduling structure's ability to identify all the possible syntactic matrices available for further constituent attachment before the constituent is actually constructed. After it is constructed, we want to be able to ask the semantic component to rank these alternative attachment points by how good the various (partial) matrix interpretations continue to be. This is a form of selective modifier placement [72] that does not require extra machinery on the part of the parser.

4.5.3 Communication lines

PSI-KL-ONE's communication with processes further down the line is also in a very rudimentary state with respect to embodying an incremental process. Currently, it does not provide any information to later processes until it has its entire input from RUS. However, the target language description produced by PSI-KL-ONE does allow for abstract descriptions of an utterance's quantificational structure, given that this structure may not be resolvable by the semantic component alone.

Since the intermediate results produced in PSI-KL-ONE are represented in KL-ONE, the same language as is used by later components, it is possible that intermediate stages could be made available to those components. These stages would allow semantic processing to receive feedback from discourse processes, as well as to improve the speed at which the interpreter could furnish information or requests to processes further down the line, such as a discourse modeler [68], a focus machine [61], a plan recognizer or a response generator. (Such pronouncements or requests might include "There's a definite pronoun filling this case - any suggestions?" .....).

4.5.4 The impact of RUS and PSI-KLONE on the rest of the system

In this section we have indicated how RUS and PSI-KLONE play two important roles in the NLU system:

- they provide an efficient, practical means of converting

the user's linguistic inputs into the KL-ONE representation of semantics that is the basis for the processing performed by the rest of the system,

- their interaction provides a model for the architecture of the system as a whole.

Our goals for RUS and PSI-KLONE stem from these two roles. The first line of work we plan to follow will make RUS/PSI-KLONE an even more effective interface by (among other things):

- making it easier for people other than linguists and programmers to extend the range of vocabulary and semantic interpretations which can be handled by the system,
- extending the range of grammatical phenomena covered by the RUS parser, and providing clear documentation of that coverage,
- making it possible for the system to provide reasonable responses to input which contains grammatical errors, unknown vocabulary or unknown grammatical constructions,
- improving the efficiency of the parsing/interpretation process by allowing the semantic component (and possibly various discourse level components) to provide preference information that will determine the order in which parses are generated.

The other major line of research will be to lay out a more complete architecture for the entire NLU system based on the ideas of incremental recognition and continuously available output described in this section.

4.6 User Interface for Input for the Dictionary System

by M. Bates

During this year a new facility has been developed to aid in

the building and maintenance of dictionaries for the RUS parser. This package has the following features:

- The user does not have to be familiar with the internal structure of the dictionary entries to define or edit the definitions.
- In most cases, the user does not have to know the meaning of the grammatical terms used (e.g., past participle, passive) because explanations are available.
- Multiple dictionaries can be maintained and used at the same time, thus allowing "patch" files and special purpose entries to be used with a standard dictionary without modifying the standard dictionary file.
- Dictionary entries are maintained on hash files. This means that the entries are loaded into core only as they are actually used, allowing the development of much larger dictionaries than would be possible otherwise.
- Wherever possible, user input is constrained to be one of a small set of legal information or is checked to ensure that it is of the proper form. This constraint minimizes the possibility of creating incorrect entries.
- An analysis package can be used to print a listing of information about the entries in a dictionary as well as the entries themselves. For example, a list of verbs that do not take complement structures is given, so that one may easily scan the list for errors.
- Users can define new syntactic categories and features while making dictionary entries, a feature that allows a great deal of latitude in the information associated with any word.

In addition to the dictionary package, we have begun to develop a semantic rule entry procedure that will take information from the dictionary, add some of the concepts and roles necessitated by the entries to a KL-ONE network, and then interact with the user to determine the additional KL-ONE structures needed to complete the representation of the semantic rules.

4.7 Dictionary Changes

The main dictionary in use by the RUS system had its origins in the Lunar system. We have removed many of the terms specific to geology (for example, BE and AS, which stood for beryllium and arsenic), and we have added a large number of new words. The new words fall into three categories: closed class words (prepositions, pronouns, etc.), frequently used English words (about the first 1000 from Kucera and Francis' book Computational Analysis of Present-Day American English, Brown University Press, 1967), and words encountered in sentences used to develop the parser (for example, the verbs in Quirk and Greenbaum's book A Concise Grammar of Contemporary English that take complement structures. The dictionary now has about 3500 entries).

The goal of the dictionary development, which is nearly achieved, is to provide a basic dictionary with all of the closed class words of English and enough of the common words in other categories to (1) provide enough data to test all facets of the grammar and (2) permit users in any domain to use the RUS system without having to define more than a few words per sentence.

The following section provides detailed documentation of the dictionary input and maintenance system, followed by a transcript illustrating the system in use.

4.8 Dictionary Package Documentation

The dictionary package allows a user to create, edit, and save dictionary definitions for the RUS parsing system without having to know the details of how the definitions are represented. It also allows words to be filed on hash files to save space in core and still allow efficient access.

Defining Words

The function MakeNewWord[word] is used to create the dictionary definition of the given word via an interactive question/answer protocol. This function is invoked when it is called directly by the user or when the parser encounters an unknown word in its input sentence. The word is always upper-cased, even if the user presents the argument in lower case.

If you type a ? to most of the prompts in the dialogue, a list of legal inputs or an explanation of what is being requested will be printed.

The following questions are asked about all words:

- "Part of speech:" The options here are as follows:
  - HELP will print some information explaining the choices.
  - NEW-POS means a new part of speech not already on the CATEGORIES list.
  - LIKE-OTHER-WORD allows you to enter a second word. The word being defined receives a copy of the definition of that word. This is useful even if

the definitions aren't exactly identical, since it might be easier to edit the copy than insert the complete definition from scratch. The second word need not be in core, and it does not have to be typed in upper case. (All parts of the dictionary package, unless otherwise noted, try to read definitions from dictionary files when needed and automatically upper case all user input.)

- NONE means that none of the current CATEGORIES apply. (There is a category called SPECIAL that should be considered instead of NONE, if you want to make that distinction.)
- The other possibilities are the parts of speech in the list CATEGORIES.

For some parts of speech (e.g., SPECIAL, CONJ, NPR) nothing more that has to be specified. For others (e.g., N, V, PRO) a specific sequence of questions prompts for further information about the category. If you are not sure what the questions mean, type a ? for help.

For adjectives, the choices for the type of adjective are the following:

- MORE-MOST means that the word being defined is a root that is inflected by preceding it with the word "more" or "most"
- ER-EST means that the word is a root that is inflected by adding "er" or "est"
- R-ST means that the word is a root that is inflected by adding "r" or "st"
- *ER-*EST means that the word is a root that is inflected by doubling the last letter and adding "er" or "est" (like BIG)
- IRR means that the word is the root of an irregular adjective (like GOOD); you will be prompted for the comparative and superlative forms

- INFLECTED means that the word is an inflected form of an adjective (e.g., BETTER or NICER); you will be prompted for details.

For adverbs, you will be asked only if the word is a negative adverb (e.g., BARELY)

For each noun, you will be asked its type.. The noun types are:

- S if the word is a root that pluralizes by adding "s" (like BOY)
- ES if the word is a root that pluralizes by adding "es" (like BOSS)
- MASS if the word does not pluralize (like MILK)
- IRR if the word is a root with an irregular plural (like CHILD)
- INFLECTED if the word is a plural form (e.g., CAVES, CHILDREN); you will be prompted for the other information associated with the word.

For prepositions, you will be asked only if the word is a bare preposition, that is, one that can appear without an object noun phrase, such as UP and HERE.

For pronouns, you will be asked for the number (singular, plural, or both) and the case (subject, object, or both).

For verbs, you will be asked the type of verb. The verb types are the following:

- S-D if the word is a root inflected like GAUGE
- S-ED if the word is a root inflected like LEARN

- ES-ED if the word is a root inflected like KISS
- IRR if the word is an irregular verb like GO, WENT or SEE; you will be prompted for the root of the verb and its conjugation
- INFLECTED if the word is an inflected form (like GIVES or HAD); you will be prompted for the other parts.

When this information has been collected about a verb, information about the features of the verb will be gathered. Every verb must be transitive or intransitive (or both); transitive verbs may or may not take a direct object and may or may not be made passive. Explicit questions are asked for each of these features. If the verb takes particles, they may be either regular particles or immovable particles. Finally the system will ask "Does it take any complements?" If the answer is y, you will be allowed to choose one or more features from a list. (You can see the list by typing a ?.) In the future, more structure will be provided here, as some of the features aren't really complement features and some are dependent on others, but for now you can choose any subset. When you have finished entering features, type END to the > prompt.

When the information specific to each category is complete, the system continues with general questions:

- "Does this word have any more parts of speech?" If you type y you will be prompted for the next part of speech and its related information. As many categories as you wish can be entered this way. When you answer n, it will go on to the following questions. If you answer Quit, it will skip the following questions.
- "Any compounds?" asks whether the word is the first word

in any compound phrase that ought to be condensed into a single word. If you answer y, you will be prompted for the list of words forming the compound phrase; it doesn't matter whether you include the word being defined in this list or not. Then you will be asked for a single word to replace the phrase. Because one word might participate in several compound phrases, you will be asked "Any compounds?" until you answer n.

- o "Any multiples?" asks whether the word starts any multiple word phrases that ought to be replaced by another multiple word phrase. That is, multiples are like compounds but result in a substitution of a list for a list rather than a word for a list. If you answer y you will be prompted first for the original list and then for the replacement list, and the "Any multiples?" question will be repeated until you respond n.

- o "Any substitutions?" asks whether another word or a list of words should be substituted for the word being defined whenever it is encountered in the input string. If you answer y, you will be prompted for that list. Because one word might have several possible substitutions, you will be asked "Any substitutions?" until you answer n.

At this point, the definition of the word will be printed, followed by the question "Do you want to edit the word?" An answer of y will put you into the word editor (see below) to allow you to correct the definition of the word.

Finally, the function asks whether you want to file the word on a dictionary file. It will list the dictionary files that are open, with parentheses around the file(s) in which the word already resides. If you specify that you want it filed, the word will be added to the dictionary file immediately (see the Dictionary Files section below).

Seeing Words

Several functions can be used to see the definition of a word, part of the definition, or whether the word is known to the system.

PrintWord[word] prints the definition of the word on the primary output file (usually the terminal). It loads the word from a file if necessary. (Whenever a word is not in core when it is needed, the DICTFILES list is searched, starting with the most recently opened file, until the word is found. See the Dictionary Files section below.) If the word is undefined, it prints a message to that effect.

MAKEKNOWN[word NoRequestFlg] is a predicate used to test whether the definition of the word is available to the parser. If the definition is already in core, it returns T immediately, otherwise it tries to load the word from a dictionary file. If it succeeds, it returns T. If it fails and NoRequestFlg is T, it returns NIL, otherwise it calls MakeWordKnown to interact with the user to get the definition.

Category?[word cat] is a predicate to test whether the given word can be in the indicated syntactic category. It is assumed that this function will be used primarily by the grammar, hence the word will already be in core. That is, this function assumes that MAKEKNOWN has been called on the word and it only looks for the definition in core; it will not read the word from any dictionary file.

Feature?[word feature] is a predicate to test whether the given word has the syntactic feature. It also assumes that MAKEKNOWN has been called previously.

FeaturesOf[word] returns a list of the features of the word. It also assumes MAKEKNOWN has been called.

StatusOf[word] prints information about whether the word is known or not, what file(s) it is assigned to, which file it has been loaded from, and which file(s) it should be written on if it has been edited.

Editing Words

To change the definition of a word once it has been created, use the function EDITW[word]. This is the only way to change the definition of a word. If the word is not in core at the time it is edited, the definition is read from a file; if the word is not on any dictionary file, an error message is printed. The editor will print the definition of the word and ask, "Want to Change, Delete, Add, File, or Quit?" File and Quit both indicate that you are satisfied with the definition of the word and wish to exit from the word editor, but File indicates that you want the word to be filed immediately, while Quit indicates that you do not want to file it now. Delete is used to remove one of the existing properties of the word. Add is used to add a property, and you will be prompted for the value of the property just as if you were defining that part of the word for the first time.

Change is a dangerous command, since it puts you into the Interlisp editor with the current expression set to the value of the property you have elected to change; you can make any edits you wish, but there is no way for the system to ensure that the resulting structure is legal, hence, you can create structures that will cause problems later. Use the Change command only if you are sure you understand the internal representation of the kind of property you are changing. Use Delete and Add otherwise.

Each time you make a modification to the word, the definition will be redisplayed along with the question, "Change, Delete, Add, File or Quit?"

Dictionary Files

Any number of dictionary files can be open at the same time. Each time you open a dictionary file (see below) the file name is placed on the front of the list DICTFILES. Whenever it is necessary to load a word from a file, the files in DICTFILES are tried one at a time until the word is found. Thus, a more recently opened file takes precedence over a previously opened file. The purpose of this mechanism is to allow patch files that override an existing file. You can rearrange the order of DICTFILES by editing it if you want, but do NOT delete or add any file names directly (use CloseDictFile or OpenDictFile instead).

Opening and Closing Dictionary Files

Normally, you should never have to do either of these

operations. When you assign a word to a file after calling FILES? (see below) the file will be opened automatically. Clever advice to SYSOUT remembers the status of dictionary files when a sysout is made and reestablishes that status when the sysout is restarted. If you have deleted the dictionary file, created a new version, or otherwise taken the files out of sync, the dictionary package will detect the difference and try to figure out the most likely thing to do. It will then ask your permission to use another file (or inform you that the file is no longer available) by a message that begins ***** PROBLEMS WITH HASHSET FILES *****.

Creating Dictionary Files

The easiest way to create a dictionary file from scratch is to call CreateDictFile[filename] and then call MakeNewWord[word] for each new word. If the new dictionary file is the only one open, you will be asked upon the completion of each definition if you want the word filed there. If several dictionary files are open, you will be asked to select the file on which the new file should go.

You can also create a new dictionary file after you have defined some words by calling CreateDictFile[filename wordlist], in which case the given words will be put on the file that is created. You can also create a new dictionary file after you have defined some new words by calling FILES?, which will ask you if you want to see what has changed in the dictionary. If you

answer y, the system will print all the words that have not been assigned to any file yet and will ask you if you want to say where they go. If you again say y, the words will be displayed one at a time (like functions that you have defined and not assigned to files) and you can type any one of the following responses to each:

```
] -- means nowhere; throw the word away.
CR -- means ignore for the present
LF -- means use the same response as for the previous word
filename -- means assign the word to that file; it will ask
            for confirmation if the file is a new dictfile.
```

After assigning the words to the correct file(s) this way, you must call FileDict[filename] to have the words actually stored on the file (just as you must call MAKEFILE after assigning functions and vars to files with FILES?).

MoveWord[word fromfile tofile deleteflg] can be used to move a word from one file to another. If DeleteFlg is on, the word will be marked for removal from the FromFile. The word is only assigned to the ToFile, not actually written to it until you call FileDict.

FileDict[filename] will always ask "Want a new version?". If you answer y, you will get a copy of the old version, with the additions made to the copy. If you answer n, the definitions will be added to the old version of the file, gradually making it larger and larger. Do not interrupt the program (e.g., with a control-D) while it is writing to a dictionary file; it may leave

a half-written expression on the file that makes it impossible to read.

Making a Readable Dictionary File

The dictionary files created by the system cannot be listed on the line printer or typed on a terminal. To get a version of the file that people can read, use the function MakePrettyFile[dictfilename prettyfilename cleanflg extrainfcflg]. The first argument is the name of the dictionary file to be prettified, the second is the name of the file to be made. Cleanflg controls whether the definitions are to be erased as they are printed on the pretty file; since the pretty file must be made by reading in each definition from the dictionary file, it is probably a good idea to have this set to T, especially if you are short of space. If ExtraInfoFlg is on, the system will accumulate a list of all the words in each category and all the words with each feature, and will print out this information (a sort of cross-index) at the end of the pretty file).

When you make a dictionary file using CLEANUP, you will be asked if you want a pretty file made as well. Keep in mind that since each word has to be read from the dictionary file and written to the pretty file, making a pretty file can be time consuming.

Recovering Space

The whole purpose of having words stored on a file and read in, as needed is to save space but as words are read in, the internal memory may get crowded. There are two ways to recover space in your working set.

CleanWords[] deletes the in-core definitions of all the words it knows that have been loaded in from dictionary files. It asks whether to erase words that are newly defined and hence not on any file yet.

CleanWord[word] deletes the in-core definition of the given word. Neither of these functions alters what is on any dictionary file, but both functions will destroy definitions that have been edited but not yet saved.

DeleteWord[word files] deletes the word from the file or files given.

Miscellaneous Useful Notes

WordsKnown[] returns a list of the words that currently have in-core definitions, including words that have been read from files, newly defined words, and edited words.

NewWords is a list of the words that have been defined but not assigned to any file.

EditedWords is a list of the words that have been edited. (This is not actually used by the system.)

DICTFILES is a list of the dictionary files currently open.

This concludes the dictionary system documentation. The following transcript illustrates a user's interaction with the dictionary package in defining a new word. The user's input is underlined.

```
MakeNewWord(BEAR)

Part of speech: n
  Type of noun: s
Does BEAR have any more parts of speech? yes
Part of speech: v
Is BEAR the root form? yes
  How is it inflected? irr
  Now we must conjugate the verb. First, present tense:
    I bear
    you bear
    it bears
What is the past tense form?  bore
What is the past participle (I have ...) ?  borne
What is the present participle (...ing)?  bearing
Now we will get the features of the verb BEAR.
Is it intransitive, i.e., can it appear without an object? no
Is it transitive, i.e., does it take a direct object? yes
Is it passive (e.g., He was hit by a car)? yes
Does it take an indirect object? yes
Does it take any particles? yes
  Enter particles as (particle newverb) or a list of such pairs.
  Regular particles: ((out bear/out))
  Immovable particles:
Does it take any complements? yes
  Choose all of the complement features that apply:
  >intranstocomp
  >fortocomp
  >end
Does BEAR have any more parts of speech? no
Any compounds?  yes
   List of words forming compound phrase: (bear hug)
   Single word to replace ' rase: bear-hug
   List of words forming compound phrase:
Any multiples? no
Any substitutions? no

Here is the definition of the word:

[BEAR
```

```
        FEATURES  (FORTOCOMP INDOBJ INTRANSTOCOMP PASSIVE TRANS)
        COMPOUNDS
            (BEAR HUG) => BEAR-HUG
        N  -S
        PARTICLES  ((OUT BEAR/OUT))
        V  (BEAR (PNCODE X3SG)
                 (TNS PRESENT)
                 (UNTENSED))]

Do you want to edit the definition of BEAR? no
Want BEAR, BEARS, BORE, BORNE and BEARING filed on
 DPATCH or (HASHDICT): HASHDICT
```

{The system would go on at this point to elicit the definitions of BEAR/OUT and BEAR-HUG, if they were not already defined.}

5. RESEARCH ON DISCOURSE TOPICS

In this chapter we report on research in three topics in discourse understanding: models of the recognition of speaker meaning, investigations into referent identification as a planned action, and approaches to interpreting one-anaphora. We discuss each in a separate section. These topics are especially valuable because they span the problems of (1) what the speaker wants a system to do, as stated in his utterances (2) how a system can distinguish phrases that are meant to refer from those that are purely descriptive, and (3) how a system can understand various anaphoric phrases.

5.1 A New Model of Speaker Meaning

by C. L. Sidner and D. J. Israel

5.1.1 Introduction

In speaking conversationally, people expect their partners to recognize their intentions, so that their partners will be able to respond appropriately. In the first part of the scenario in Chapter 3, the speakers encoded their intentions in a variety of sentence types. Instead of telling the hearer what to do, the speaker stated his goals, and expected at least a partial response. The sentence forms he used were not simply commands, but rather declaratives that describe a desired state or a new difficulty or comment on progress. This section presents a

model, developed by Sidner and Israel, in which recognizing the speaker's intended meaning plays a fundamental part in determining a response. First we describe our methodological and theoretical approaches to this problem, and then we describe a model that enables us to tell what the speaker's intentions are. We then illustrate the model on two examples and compare it to our previous work.

We think the model that we present here is particularly powerful because of the capabilities it provides to a system that understands natural language and presents information on a graphics display. It provides a system that can reason about utterances that express errors in planning, acknowledge those errors and respond to them just as people do in conversation.

5.1.2 Background for a new model

As part of our research to provide a sophisticated natural language, graphics oriented system (NLGO) for a decision maker, we collected protocols of users communicating with a simulated version of the system. The simulated system was actually a person communicating in English over a terminal and computer link to a user. For graphics we used an overhead viewgraph projector to project the drawings that the person simulating the machine drew. Using an arrangement reported in [62], the user was able to point at the display and make changes as needed. The protocols consisted of the transcript of communications, all the pictures drawn during a session, and notes indicating deictic

references to screen objects[33]. When we analyzed those collections of protocols, we recognized that a NLGO system would require the ability to understand the following types of declaratives: ones that describe a desire, state a problem or function as a comment on progress. To include this ability within our on-going research, we found that we would need to expand the existing pragmatics component (see [13] in two ways: (1) by recognizing a richer form of plans, and (2) by making explicit the connection between the speaker's intentions as structured by his plans and the response intended by the speaker.

5.1.3 Defining Intended Speaker Meaning

Our goal is to provide a computational model of the hearer's interpretation of the speaker's intended meaning. The intended meaning of an utterance we define as that set of <propositional attitude (e.g., belief, want, intend), propositional content> pairs that the speaker wants to induce in the hearer by means of the utterance.

The notion of the intended meaning of an utterance can be illustrated by contrasting it with that of semantic meaning. The semantic meaning of a declarative utterance is the propositional

---

[33] A deictic reference is one where a person uses a linguistic phrase to refer to something by pointing at it. The person may actually point his finger, cock his head or otherwise physically point or he may simply use a phrase that makes clear he's pointing.

content assigned to that type of utterance by the semantic rules of the language. For instance, if someone says, "You're a prince," the semantic meaning is that the person addressed by the speaker is the son of a king. By contrast, the intended meaning depends on the psychological state of the speaker at the time and place of utterance. The speaker may mean that he thinks the hearer is a really nice guy and wants to tell him so, or he may be saying something quite different. The speaker, using irony, may mean that the addressee is just the opposite of a nice guy.

This example demonstrates that the speaker's intended meaning, though correlated, is not in general identical, with semantic meaning. Comprehending the semantic meaning of the utterance forms the basis for discerning the intended meaning, but understanding the intended meaning also requires the use of the following beliefs by the hearer:

1. The characteristics of the current situation
2. The speaker's beliefs and goals
3. The context of discussion (the <u>discourse context</u>) as a special aspect of (1)
4. The conventions for action that exist between the speaker and hearer[34]
5. The mutual beliefs of the speaker and hearer concerning (1) through (4).

---

[34] This, of course, is relevant only to a special class of situations; a class that includes the kind of interaction the BBN system must handle.

A sample exchange will indicate the role of these kinds of beliefs. In the example below, the user is interacting with our NLGO system to display some information. The user's first two utterances are simple, direct imperatives that indicate that the user wants the NLGO system to display a part of the net and then move the focus to a subpart of the display.

D1-1 U: Display the clause level network.
2 S: <display of network> OK.
3 U: Now focus on the preverbal constituents.
4 S: <display of subnet> OK.
5 U: No, I want to see S/AUX.

What does the user mean by her third utterance (utterance 5)? The answer depends on what she believes about the net objects to which she has referred. Suppose she thinks that S/AUX is part of the preverbal constituents. Then she is communicating that the display is wrong and what's wrong with it; she intends for S/AUX to be included in the display with the other constituents. Suppose, alternatively, that she thinks that S/AUX is not part of the preverbal constituents. She is still indicating that she wants to see S/AUX, but also that she has changed her mind[35] about the display in some way and intends S/AUX to be visible. This discourse, similar to one discussed in [13], could be handled by our first prototype only as a modification of utterance (3) by utterance (5). Reasoning about the user's overall task and the relation of S/AUX to it was not considered.

---

[35] Either she has realized an error in her plans, or that the error is due to a mismatch between her views about S/AUX and the system's.

In choosing among its available responses, the NLGO system must use its model of the user's beliefs about the domain and its model of what the user takes to be mutually believed between the two of them about that domain. For example, the user might have thought that S/AUX was one of the preverbal constituents, and thought the NLGO system believed this also. She would then have expected and intended the NLGO system to include that state in the display. If the user had been right about this belief, the NLGO system would indeed have included it. But the user's "No" indicates to the NLGO system some bug in her plan, a bug stemming either from a faulty model of the domain itself or from faulty expectations about the NLGO system's model. [For simplicity, we assume that the NLGO system is omniscient about the ATN grammar.]

If, on the other hand, the NLGO system doesn't conclude that the user takes S/AUX to be among the preverbal constituents, and if it believes that she takes <u>that</u> idea to be mutually believed, then it must again use its models of her and of her model of itself to determine what action <u>is</u> intended by the user [E.g., should it compress the current display to make room for S/AUX; should it erase the current display and bring up a new one, centered on S/AUX, etc.?]. This decision may depend on the kinds of conventions alluded to in (4) above. In general, of course, people's behavior in conversational situations also depends on the relative status of the conversational partners, on what the participants think will benefit themselves, as well as not harm others, and the like. These social considerations are

significant to human interaction, but for the remainder of this paper, we'll assume that the NLGO system responds in a slavishly cooperative way, that is, it has no interest beyond serving the user.

There are two ways to view the intentions of another agent. The first is simply in terms of one's beliefs about what the other person wants and believes. This is keyhole recognition (see [23]). One person decides what he thinks another intends simply by observing him through a keyhole; for example, I decide that you are looking for your umbrella, on the basis of your looking around the room with your coat on, when I believe you believe that it's raining outside). Keyhole recognition of a user's wants is central to Genesereth's MACSYMA advisor [26]; it also forms the basis of plan recognition in both [56]'s work on BELIEVER system and in Wilensky's story understanding [70].

The intended recognition of what someone is doing, on the other hand, is relevant for communicative situations [30], [1]. A speaker says something to a hearer and intends the hearer to recognize the intention that lies behind the utterance. The speaker is attempting to "give the hearer a piece of its mind" and it's essential to the success of the speaker's attempt that the hearer recognize it as such. In Allen's terms, the shared recognition of intention is expressed by the speaker saying something to the hearer because the speaker wants the hearer to believe the speaker wants something. That is, H believes S wants

H to think (believe) that S wants something (HBSWHBSW). In the example D1 above, shared recognition is involved in the fact that the user wants the NLGO system to believe her statement, a statement about the user's wants. Generally the hearer, in her task of responding to the speaker, must take into account not only the shared recognition of the speaker's intentions but also the beliefs that the speaker assumes are shared with the hearer (in Cohen's model [20] these are beliefs in the context HBSWHBSW). Occasionally the hearer's own beliefs about a situation will differ from the shared beliefs (called mutual beliefs hereafter) and influence the hearer's response. Just how this occurs will be illustrated later.

5.1.4 A model of recognition of intended meaning

The hearer's task in recognizing what the speaker meant by an utterance is to be understood as follows:

1. to produce an explanation for the utterance, stated in terms of the speaker's beliefs and wants
2. to use the explanation as a basis for a response.

We use the term "explanation" because the hearer is trying to answer the question "Why did the speaker say that to me?" The answer to this question - the proffered explanation of the speaker's act in uttering what he or she did - in turn produces new beliefs about the speaker; these will form part of the basis of the hearer's response.

The explanation, in general, will have the form of a set of pairs of propositional attitudes and propositional contents attributed by the hearer to the speaker. [E.g., <belief, that S/AUX is part of the preverbal constituents> <want, that I display all components of the preverbal constituents>, etc.] Certain beliefs play a central role in explaining why the speaker said what he did:

Explanatory beliefs

1. Beliefs about the speaker's goal and the plan to achieve it
2. Beliefs about the hearer's capacities
3. Beliefs about the hearer's dispositions to act given information about the speaker's wants.

The problem we pose for ourselves is determining how to infer beliefs of these kinds and how to use them to distinguish between intended and helpful but unintended responses. We want our NLGO system to recognize and produce the intended response whenever possible, and to be able to produce a helpful response when appropriate.

To model the construction of the required explanation, we begin with Grice's theory of speaker meaning ([30] and [31]). Grice notes that there are certain kinds of evidence normally available to an audience on the basis of which the audience is intended to draw certain conclusions about the speaker's intended meaning. These kinds of evidence include (1) the features of the

utterance, and (2) mappings between those features and propositional attitude-propositional content pairs that the audience (assumed to be a competent speaker/hearer of the language) is supposed to be able to grasp, and is intended to grasp. For example, the feature: DECLARATIVE will be mapped to the speaker's wanting the hearer to believe the speaker believes the propositional content of the utterance; while imperatives will be mapped to the speaker's wanting the hearer to believe the speaker wants the hearer to bring about the state of affairs expressed by the propositional component of the utterance.

Somewhat more formally: an audience, for the utterance of a certain sentence S1, who is believed by the speaker to have certain attributes A, is expected to be able to recognize certain features of the utterance and to be able to draw from those features certain conclusions about what the speaker intended in uttering S1 in that context. [One such audience attribute, of course, is competence in the language of S1; others are both more interesting and more specific to the situation.] These conclusions include:

Intended Conclusions

1. S1 has certain features (call them F1 ... Fn).
2. S1 is correlated, in virtue of such features and the rules of the language, with the pair (p , PC(S1)).
3. The speaker intends the audience to believe that the speaker p's that PC(S1).
4. By sincerity (see below), the speaker does p that PC(S1),

5. the speaker intended that the hearer p[36] that PC(S1).

[In the above, "p" is a schematic letter that takes verbs of propositional attitude as substituends; "PC", a schematic letter that takes declarative sentences as substituends.] We can apply this theory directly to the sample dialogues. For example, let us consider a sample utterance from the dialogue D1, understood, however, as the initial utterance of a discourse:

S1: I want to see S/AUX.

Intuitively, we would like the theory to allow us to show how an audience (even a computer system) would conclude that the user wants to see S/AUX, and that the user wants it to believe that he/she has this desire.

The set of relevant features F, attributes A and mappings C include:

- o F1 = S1 is in declarative mood
- o F2 = S1 was uttered intentionally by U
- o F3 = S1 was intentionally directed at S
- o A1 = S is a computer system with a graphics display, and U knows this
- o A2 = S believes U is sincere

---

[36] Actually the hearer may be intended to have a different propositional attitude p' toward a related proposition. For simplicity, we'll assume these are the same.

- o C1 = F1 maps to U's wanting the intended audience to believe that U believes that U wants to see S/AUX.

Our proposed system will make default assumptions guaranteeing F2, F3, A1, and A2, recognize that F1, and apply C1 to S1. The NLGO system can then use the intended conclusions and infer directly that:

1. U intended (S to recognize) that S1 is correlated with U's wanting S to believe that U believes that U wants to see S/AUX (derived from intended conclusion 3 and C1).
2. By sincerity: U believes that U wants to see S/AUX.
3. By reliability: U wants to see S/AUX.[37]).

This, of course, is what, on intuitive grounds, we wanted the NLGO system to conclude.

5.1.5 Extending the model

While the Gricean framework provides a starting point for recognizing the speaker's communicative intentions, it does not provide a recipe for inferring the intended response. Given, for example, that the user wants the NLGO system to believe that the user wants to see S/AUX, and nothing more, the system could

---

[37] Simply stated, these rules, for the case of belief, are: Sincerity: If x wants y to believe that x believes that q, then x believes that q. Reliability: If y believes that x believes that q and that x is reliably informed about q, then y will believe that q. The basis for these rules is the intuition that the speaker is sincere about his beliefs, and that what he believes he believes reliably, at least for certain subject matters, such as his own present state of mind.

simply say "Yes, I understand," (or "Let me add that to my data base of beliefs about you") - a behavior the user probably did not intend. At the same time, the NLGO system could decide to provide a lot of information by showing the whole ATN network and highlighting S/AUX. Such behavior might even be helpful; but it is not, we can presume, the intended response.

To determine the response the user intended, the NLGO system must consider the utterance in a larger situational context. This context is determined by what (it thinks) the user is doing, what (it thinks) the user thinks the NLGO system can do, and how cooperative (it thinks) the user takes the NLGO system to be. We now turn to a description of a method for inferring the intended response from the initial intended conclusions about the user's beliefs and desires.

We have augmented the Gricean framework to enable the NLGO system to derive a situation-specific explanation for the user's having the wants and beliefs he or she is believed to have. In particular, the system can be viewed as asking itself for an explanation of some of the beliefs it attributes to the user. The explanation is of the same type as that given earlier. For example, to explain why the user wants the system to perform some action, the system would infer that the user is pursuing a plan in which that action is a step. [This process must stop somewhere; in fact it stops because some plans are simply assumed to be entered into for their own sake and to require no further explanation.]

An example will illustrate what we have in mind. For utterance S1 above and the conclusions about the user's wants regarding S/AUX given previously, the NLGO system seeks to explain why the user wants the NLGO system to believe that the user wants to see S/AUX, and perhaps why the user wants to see S/AUX. To answer the first question, the system determines if any of the plans it has provisionally attributed to the user contains this step; and if so, it determines what relevant capacities the user believes it to have. For the case at hand, since there are many such plans (deleting S/AUX, rearranging its arcs, etc.) and since this is the initial interaction, no detailed plan-information is deducible. The user is assumed to believe, however, that the NLGO system has capacities relevant to seeing S/AUX--e.g., displaying it on the screen. The system concludes that the user intends this capacity to be used, and since the system is cooperative, the system produces a display. An explanation for the user's wanting to see S/AUX may not be forthcoming. (It may also not be needed for the system's response planning.)

This extended theory depends not only on the Gricean framework but also on the ability to create an explanation based on the user's plans. This last involves:

1. Recognizing the correlations between utterance features and pairs of propositional attitudes and propositional content
2. Using the properties (P) of the NLGO system (described above in Grice's theory)

3. Determining the goals of the user from the propositional attitudes, where the goals are structured in a hierarchy of goals and subgoals

4. Deciding on the capacities of the NLGO system (mutually believed to be capacities) that are relevant to the speaker's goals

5. Using the speaker's recognized goals as an expectation model for the remaining part of the discourse.

To implement this model, we are using a number of available AI tools (the implementation is not complete). The NLGO system must have definitions of a number of plans, so we are using Sacerdoti-based procedural networks of plans [55]. Beliefs and wants must also be represented, and for this we are relying on Allen's and Cohen's models of belief and want contexts. A crucial aspect of this model is a method of "parsing" the user's wants as steps in plans; we are currently studying algorithms using an ATN formalism, but modified to allow for bugs in a plan, recognizable with a small bug library (see [65]). This model bears some similarity to Genesereth's plan recognizer [26]; it is distinguished by recognizing a different class of bugs than Genesereth's. This method makes it possible to use a plan, once selected from the collection of plans by unique substeps, as an expectation device for the remaining part of a discourse. Finally, we use standard antecedent reasoning for deducing the correlations between utterance features and propositional attitudes, and for relating user plans and the NLGO system's capacities.

5.1.6 Reasoning about a user's "buggy" plans

In the previous discussion, we have shown the utility of explanations reflecting (among other things) beliefs about the speaker's goals and about his beliefs about the NLGO system's capacities. Now we will demonstrate what additional reasoning such a model enables. In particular, we will show that such explanations provide a NLGO system with a means of discerning bugs in a user's plan. We will show how declarative utterances can be used to indicate bugs in the user's plan, problems that are stated in terms of what the user wants or expects or in terms of actual descriptions of difficulties in proceeding to the next part of what the user plans to accomplish. The model of speaker meaning presented so far makes it possible to interpret such utterances.

In the first example, the user is unaware of a bug in his plan. The NLGO system, after recognizing the bug, must inform the user, because no satisfactory response is possible until the bug is resolved. In the second example, the user discerns a bug and informs the system; awareness of this bug allows the system to recognize the intended meaning of a subsequent utterance. We will present enough detail of each example to permit the reader to see how a program could embody this reasoning process.

Let us return to example D1 given below.

D1-1 U: Display the clause level network.
2 S: <display of net> OK.
3 U: Now focus on the preverbal constituents.

4 S: <display of subnet consisting of S/Q, S/HOW, S/, S/QDET, S/NP, S/DCL.> OK.
5 U: No, I want to see S/AUX.

After the request to focus on the preverbal constituents, the NLGO system recognizes that the user's plan involves examining the preverbal constituents:[38]

```
Examine User "preverbal constituents"
        Cause User
         (Display System "preverbal constituents")
        See User "preverbal constituents"
```

The plan has two steps; the first is to cause the system to perform a display, and the second is to look at the displayed items. [Plans typically have preconditions; steps may be primitive, or may be composed of actions, requiring other plans as well.]

Following the last utterance of D1, the system can infer that by "no," the user is signifying that his plan has failed in some way. In interpreting the rest of the sentence, the system will reason about the user's intentions [as shown previously] and conclude that the user's intention is to see S/AUX. This intention nearly matches step 2 of the plan deduced. It differs because S/AUX is not a part of the preverbal constituents. Using a small bug library, the NLGO system will recognize a possible bug in the Examine plan.

---

[38] We assume here that the phrase "preverbal constituents" is interpreted appropriately, but will not discuss this interpretation here.

To account for the bug, the system can reason in either of two ways[39]. On the one hand, if it now has reason to believe that the user believes that the preverbal constituents include S/AUX, it will conclude that there is a bug in the user's plan. This is a private (not mutual) belief; but it prevents the system from responding in the way intended by the user (to display S/AUX) for not enough of the user's intentions are clear to decide how to do the display (e.g., to include S/AUX or to show it alone). Hence the system will respond by indicating what the bug is and by asking about the particular mode of display desired.

On the other hand, if the system believes that the user believes the preverbal constituents are (mutually believed to be) disjoint from S/AUX, then the system will conclude that the user has scrapped his current plan, and that this conclusion is one the system is intended to deduce. In this case, displaying S/AUX is the intended response, but the NLGO system must still ask how to display it, since it is not clear whether the user intended it to be displayed alone or with the subnet. A person, in such circumstances, would probably conclude that S/AUX should be displayed alone, because s/he could deduce that in general if a plan is scrapped, effects of its partial realization are no

---

[39] Actually there is a third case--the system was wrong about what "preverbal constituents" refers to. As mentioned above, we ignore this case.

longer desired. However, this heuristic may be too general for systems that still have limited reasoning capacities, and hence we have chosen not to include such rules.

This example demonstrates two aspects of our NLGO system: its use of plan-attributions, inferred in the course of interpreting the user's intentions, to recognize bugs in plans, and its use of private as well as mutual beliefs to determine its response when what the user intends is unclear because of a "buggy" plan.

The next example concerns the first part of the NLGO system scenario where the NLGO system is interacting with a user when a graphics display is available for representing information about a database.

D2-1 U: I want to see the Generic Concept named employee.
2 S: OK. (displays concept in mid-screen)
3 U: I can't fit a new Individual Concept below it.
4 Can you move it up?
5 S: Sure. (moves up the Generic Concept)

To respond to "Can you move it up?" the NLGO system must determine whether the user meant his utterance directly as a question about the system's abilities, or whether the user intended to direct the system to move the concept under discussion. Its decision depends on inferring the speaker's plan and, in particular, on what it believes the user's model of its own capacities to be.

This example illustrates a feature of natural interchanges:

a user may have a plan in mind, and carry out a part of it, without considering possible undesired side effects; when one occurs, it may be recognized and eliminated. In D2 the user is carrying out the plan of accessing the Concept for EMPLOYEE so that she can add a new employee to the database. She wants the system to display the EMPLOYEE Concept, but has not foreseen that its display location might be inappropriate. After the inappropriateness is discovered, the user indicates the difficulty and expects it to be corrected. Just how the bug in D2 is corrected depends on whether the user already believes that the system can move things up and intends the system to do so, or whether she has to find this out first.

From the NLGO system's point of view, the decision about what the user means may cause it to respond differently in various cases. Suppose the system thinks the user believes that the system can move up concepts on the screen. Then when the user indicates that his plan has a flaw (D2-3), the system must conclude that the user's plan is blocked by the lack of space for a new concept. When the question about moving the employee concept is raised, the NLGO system will conclude that the user intends to tell the system to perform the move by asking about a precondition of the action she wants, a precondition that consists in the system's having a capacity it is mutually believed to possess. The system is intended to move up the concept, not simply to answer the question.

A different scenario is as follows. Suppose the NLGO system thinks the user is unaware that the system is capable of moving up the concept. Then, when the user indicates that his plan has a flaw and asks about moving the employee concept, the system will conclude that the user intends to find out whether it has that ability, as part of finding a means of resolving the block. In this case, if the system moves the concept, that is a bit of helpful behavior, one not intended to be recognized as intended by the user.

We will outline in some detail how our NLGO system reasons in such contexts by showing what plans are deduced, what rules are needed, and how the reasoning proceeds in the case of D2-3 and D2-4. The relevant user plan is:

```
Add-Data <User> <netpiece> <data> <screen-location>
        Consider-aspect <User> <netpiece>
        Put User <data> at <screen-location>
```

The Add-Data plan states that to add data, a user must consider some aspect of a network part (netpiece) and then put some data at a screen location. Even after recognizing that the user wants some data displayed from D2-1, the system cannot deduce that add-data is the user's plan. Since there are many ways to consider some aspect of a net (ask for a display, think about it, ask to be informed about its contents), as well as many other plans for which displaying a netpiece is a first step, the user cannot be understood to have intended the system to recognize that his plan was to Add-Data . All the system can

conclude is that the user wants the employee concept displayed, and it r ponds accordingly.

In reasoning about D2-3, "I can't fit a new Individual Concept below it," the NLGO system concludes that among the speaker's intentions mutually presumed to be recognized is that the user produced a declarative utterance with the propositional content that the user cannot fit a new Individual Concept (abbreviated e2) below the Generic Concept (abbreviated e1):

```
BELIEF1  (Say User System (Declarative
           (Not (Can User (Fit User e2 (below e1))))))
```

From this, the system concludes that the user wants the system to believe that the user believes that it can't fit e2 below e1, and that the user in fact believes that he can't. The system then infers the embedded proposition [(Not (Can ...))], and that the user intended that that proposition be mutually believed. Using a (default) rule to the effect that whenever a user says that it can't bring about a certain state of affairs or perform a certain action, the user is telling the system that it wants that state of affairs brought about, the system concludes that it is intended to believe that the user wants it to believe

```
WANT1:(Fit User e2 (below e1)).
```

The system seeks a partial explanation of this intention. It decides that the previous request for a display of the generic concept, together with the inferred intention to fit, are good evidence for the add-data plan as the intended plan.

Now the system can bring to bear a rule to the effect that if it believes that the user has informed it that he or she can't perform a certain action which he or she wants to perform as part of some plan he or she is pursuing, then it should conclude that the user intends that that action should be unblocked. Whether and how the system is expected to respond depends upon whether the system believes that the user believes there is some action available to the system relevant to this unblocking. In fact, the NLGO system might have several relevant capacities (such as moving up screen objects or erasing the screen as alternative ways to make room). Where the system believes the user knows this and hence concludes that the user wants to exploit some system capacity, it must await further information to determine which action was meant. In any case, on the basis of attributing the unblock plan, it can interpret the user's question as a way of bringing about a move-up action rather than simply a desire for information. Even if no such request as "Can you move it up?" were to follow, the NLGO system would have a basis for asking about the user's intent ("Do you want me to move up the concept or empty the screen?").

If the NLGO system believed that the user was unaware of its capacities to move screen objects up, it would reason no further on D2-3 (again, because it has not recognized any intention on the part of the speaker that it act). D2-4 allows the system to deduce that the user wants to know if it can move el up, since yes-no questions are taken as signalling intentions to know if.

The system, in seeking an explanation, can conclude that this is the first step in finding an agent with a certain capacity, and that this action is a means of unblocking the step of putting objects on the screen. The system must respond to the user's intention by telling the user it can move up the display, but the choice actually to move is made as helpful behavior[40] because no intention that the system move anything up has been recognized.

In both of the above cases, the plan to Add-Data to the screen is known to be in effect. Hence once the bug is cleared away, the NLGO system is prepared to interpret subsequent utterances in light of this plan. Since the user's subsequent utterance in the proposed scenario requests that an Individual Concept with a Role first name "Sam" and last name "Jones" be put on the screen, the system would recognize this request as part of Add-Data and determine the intended location for the IC on the screen--below the Generic Concept for EMPLOYEE. In this way, the plan becomes an expectation device for the next portion of the conversation.

This example illustrates not only that mutual beliefs about the NLGO system's capacities affect the system's determination of the user's intentions, but also that the full explanation of each

---

[40] based on rules about helping out when one has an appropriate capacity. As with other heuristics, it may be wise to monitor carefully what the system does to be helpful, since some helpful actions, if not in fact desired by the speaker can be easily undone, but others have serious side-effects.

utterance deepens the system's understanding of the user's goals and subsequent utterances.

5.1.7 Comparison with an earlier approach

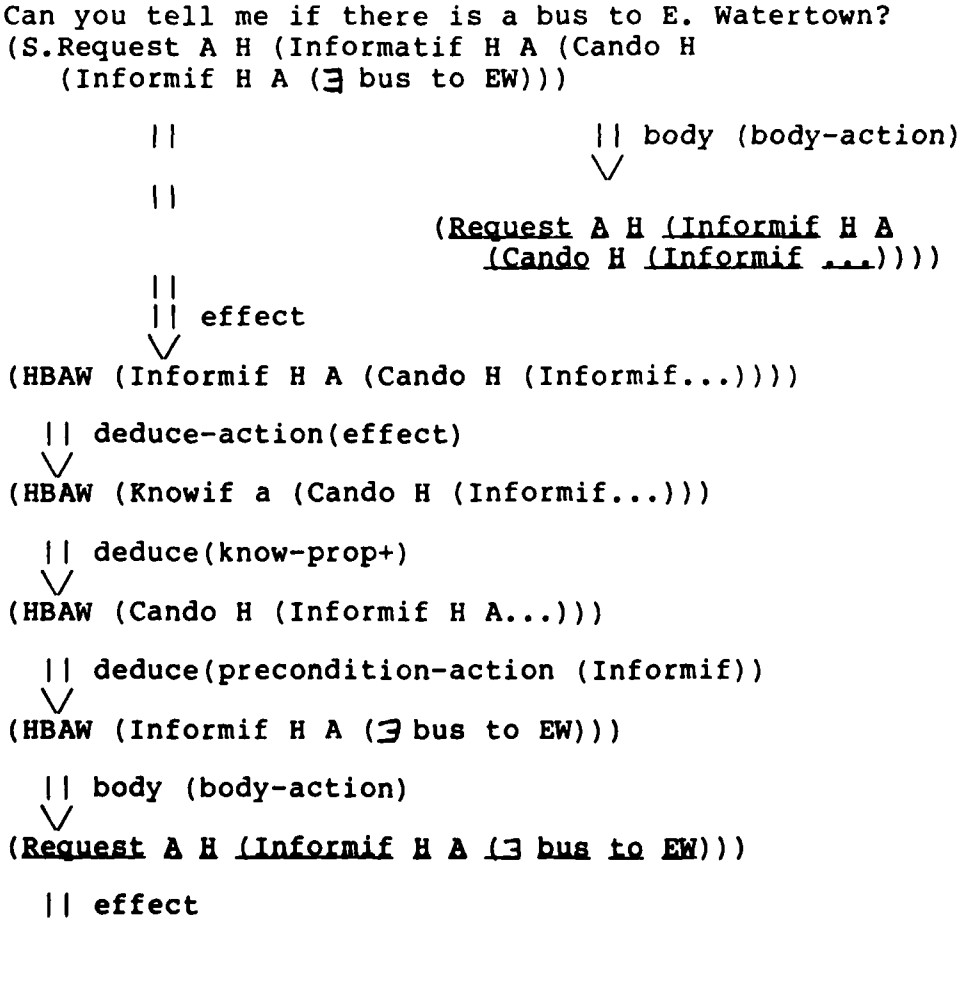
In this section, we present one short example of how Allen's algorithms, which form the basis of our previous work on speaker meaning, proceed on utterances from the domain of information about trains in [1]. We will show how our proposed model would proceed and discuss the differences in the two approaches. As we will illustrate, a crucial difference is using the knowledge of what the speaker is doing (that is, his plan) and knowledge of the hearer's (that is, the system's) capacities to determine a response. Allen's model relies largely on invoking general rules true for any action; while our model uses some general rules about actions, it also brings to bear its knowledge of particular actions the speaker is executing and its knowledge of its own beliefs as a hearer about its capacities. Because the model brings these kinds of knowledge to bear, it explicitly connects the speaker's intentions, expressed as plans with the response intended by the speaker.

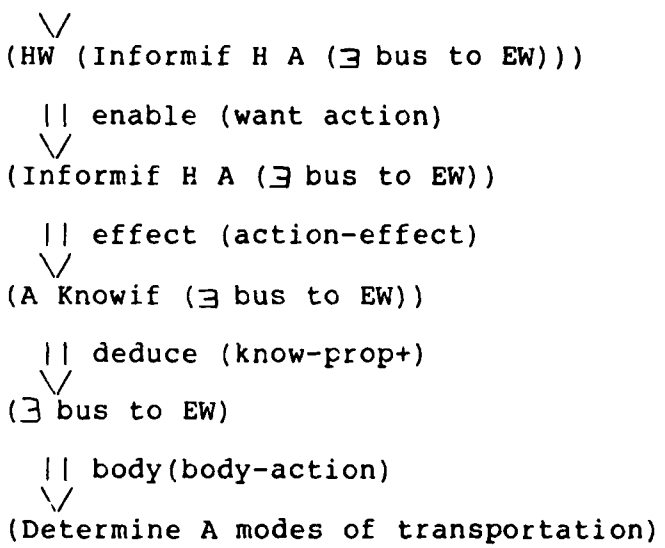
Figure 13 illustrates the processing of Allen's model for the utterance, "Can you tell me if there is a bus to E. Watertown?" Notice that the algorithm assumes that this question maps to an s.request[41] by actor A to hearer H of an

[41] for definitions of s.request and informif, see [1].

informif by H to A of whether H can do an informif of the existence of a bus to E. Watertown. The reader should observe that two speech acts, both requests, are identified during the processing (they are underlined in the figure). The model, after deducing all the propositions (called the plan), uses them to determine obstacles in this plan, obstacles it should overcome. One obstacle it might recognize is: (A Knowif (∃ bus to E. Watertown)). This obstacle can be overcome by an informif.

FIG. 13. SAMPLE PROCESSING FOR ALLEN'S MODEL.





By comparison, our model will reason as follows. For the question "Can you tell me if there's a bus to E. Watertown?" H will ask itself why A said that to H and will conclude on the basis of "can you" questions that:

> Want0: A wants to know if I (H) can tell him (A) if there's a bus to E. Watertown.

Such a want causes the system to ask itself for an explanation of that want. To do so, H uses a rule that states that if someone wants to know something and asks a question to a person who is believed able to answer that type of question, then an answer is wanted, i.e.,

> Want1: A wants H to tell A if H can tell A if there's a bus to E. Watertown?

Now H asks why does A want this? H turns his attention to what A is doing and deduces A is a stranger in a train station. Such

people, H reasons, are usually interested in modes of transportation, such as trains and buses, and for getting to places. To find a means of travel one can look at a travel book or ask a person who has knowledge such matters. If one chooses to ask a person, then one must identify a person with such knowledge. H believes that he is one such person and W0 indicates that A has decided to find out that H is. Hence H is able to conclude that A needs information about modes of travel, rather than just information about H's capacities. Then, by looking at the internal structure of Want1, H can determine just which modes of travel A wants to know about, and conclude that A has Want2:

Want2: A wants me to tell him if there is a bus to EW.

Before going further with H's reasoning, it is important to see that the particular situation of the speaker and hearer are significant to the deductions the hearer can make. The inference to Want2 depends on the fact that both speaker and hearer know that A is a stranger in a train station and that as such a person wants to know about transportation modes. Furthermore, the capacities of the hearer, which are mutually known to hearer and speaker, help the hearer explain what the speaker says. Hence H is not merely good at guessing what A wants, but rather A intends that H conclude Want0, Want1, and Want2.

Just as with Want0 and Want1, H must ask himself why A wants Want2. Here the answer brings to an end any further

explanations. A wants information about buses (call that Want3) but H cannot draw any further conclusions (such as answer to why A wants Want3) because such conclusions would involve guessing about A's wants and desires. There is no reason to believe that such guessing is intended by A, so H suspends reasoning further about A's wants.

Now that H has established several of A's wants, we can ask, what does H do with his explanation? Two of A's wants (Want1 and Want2) involve H as the agent of an action wanted. H brings to bear knowledge about what A knows of H's capacities to determine that A intends for H to act on these. For the other wants, Want0 and Want3, H may act, as a helpful person, in some way H believes appropriate, but his actions are not intended by the communication. In carrying out Want1 and Want2, H must reason that if he performs the telling of Want2, he will also perform the one for Want1, so that only one action is necessary.

The major difference in these two models is use of the speaker's plan. Allen, who takes all the conclusions inferred in his model to be part of the plan, does use the speaker's overall goal in reaching those conclusions. Instead, that goal is the last conclusion reached in his model. By contrast, in our model the speaker's overall goal (to determine a mode of transportation and find someone who knows about it) is used early in the processing, in addition to rules about the reasons for actions.

The two approaches also differ on what the hearer takes the

intended response to be. For Allen's work, the response is the overcoming of obstacles that another component of his model can identify. Overcoming these obstacles is taken to be the intended act because in general the speaker wants the hearer to be as helpful as possible. In our model, helpful behavior is distinguished from more directly intended responses such as Want2. These responses must be explicitly related to the speaker's intentions, viewed as a plan. In addition, our model takes into account specific capacities of the hearer in determining an intended response.

A final distinction between the two models concerns the utterance and initial conclusions drawn from it. Allen assumes that utterances can be mapped into surface speech act forms (such as s.request) for his model. On the other hand, we have chosen rules that map from the utterance to the NLGO system's first conclusion by conventions about how English is used. The type of conclusion drawn is not one that identifies an act that A wants H to perform, but rather states some want that A has about his own concerns.

One difference between the two approaches cannot be easily illustrated in the domain of information about trains. We have expanded on the notion of plans originally given in Allen's work to include multistep and interrupted plans. Allen's model cannot recognize these kinds of plans because the model's plan language is not equipped with a means of describing them.

In summary, we have presented in this section a model of the interpretation of speaker meaning which takes into account several different kinds of belief: about the current situation of the speaker, bout the speaker's goals, about the discourse context, about the speaker's knowledge of the hearer's capacities, and about the hearer's conventions for acting. Our model, both in the abstract and in its computational form, infers an intended response on the basis of these beliefs by producing an explanation for each of these beliefs and using that explanation for further conclusions about speaker intentions. This model makes possible the recognition of the speaker's intended meaning not only of imperative and interrogative utterances but also of declaratives that serve as comments, complaints, checks on progress, and announcements of bugs in the speaker's attempts to achieve his goal.

5.2 Utterance Planning

by P. R. Cohen

5.2.1 Deciding what to say

Current natural language systems have great difficulty in producing appropriate responses. The difficulty stems not so much from an inability to generate sentences as from an inability to decide what to say. Given a delimited propositional content, algorithms can be developed to produce isolated natural language utterances that express that content. However, there are as yet

no algorithms for determining just what facts and properties should be expressed in a given context.

The problem of deciding what to say manifests itself in many ways, for example:

1. Deciding what communicative act to perform -- a request, question, informative statement, suggestion, etc.
2. Deciding what the content of that act should be -- what is to be requested, questioned, suggested, etc.
3. Deciding how to refer to entities -- should a definite or indefinite noun phrase or a pronoun be used? If a noun phrase is to be employed, which descriptors should be expressed?
4. Deciding on overall utterance form -- Should the utterance be a complete sentence or a fragment (noun phrase, prepositional phrase)? If it's to be a sentence, should it be a declarative, interrogative, or imperative? (Note that this decision is not identical to, but is related to, deciding on the communicative act to be performed). What should remain in focus or be introduced into focus?

We have been investigating, with the work of Cohen, the role of reference planning in task-oriented dialogues in order to answer some of these questions (see [21]).

As an illustration of the need for explicit reasoning about referring phrases, consider that the answer to a simple question ("Where does John live?") should depend upon the respondent's knowledge of the questioner's beliefs and plans. If the respondent believes the questioner plans to drive to John's house, it might produce a route description, or supply information necessary for the questioner to identify John's

house. If, however, it believes the questioner intends to send John a letter, it should supply a street address and zip code. If the questioner is believed to be taking a survey of the geographic distribution of a set of conference participants, an appropriate answer might include a state name (California) but not a country (since the questioner is believed to know the country already). Thus, a response planning component must be sensitive to the beliefs and (especially) the plans of the user.

5.2.2 The Act IDENTIFY

Searle [59] has claimed that the communicative act of (singular definite) reference involves uttering an expression D with the intention that the hearer pick out, or identify the referent of D. From the perspective of a plan-based theory, choosing a phrase that enables a hearer to identify a referent should be represented as an action in the speaker's plan. As such, it can be reasoned about just like any other act.

Identification is essentially a search process, the act of searching for something that satisfies the description. We can approximate its definition with the following:

IDENTIFY(agt, D, x)
(where agt is the agent, D is a description)

precondition: There is an object x perceptually accessible to agt such that x is the referent of D.

effect: (IDENTIFIED-REF agt D)

means: Some function mapping the description D to some procedure that when executed yields a (perhaps) "direct representation" of the referent of D. That procedure may well incorporate the results of perceiving the world.

What is not clear, of course, is just what agt has to know about D to say he has identified it.[42] To give a name to the state of knowledge one is in after having identified the referent of D, we will use (IDENTIFIED-REF agt D). We shall only consider the "basic" case of identification through perception.

C. The problem areas

There are (at least) six problem areas that need to be addressed.

1. Is there evidence for an IDENTIFY act in speakers' and hearers' plans?
2. If so, why is it planned?
3. What advantages accrue from positing such actions? Do they allow us to develop a uniform analysis for what would otherwise be unrelated phenomena?
4. What signals or communicates a speaker's intention that a hearer identify the referent of a description?

---

[42] Use of a standard name or rigid designator in the possible worlds framework implies agt is so "acquainted" with the referent of D that he could not possibly misidentify it. That, however, is too strong a condition. Ultimately, one would like to make identification relative to a "purpose".

5. When do speakers explicitly communicate the intention that their hearer identify a referent?

6. How is IDENTIFY defined as an act? In what situations is it appropriate? What changes of "state" does it produce? By what means is it accomplished?

We have been investigating problems 1, 3, and 5 by exploring evidence from task oriented dialogues.

5.2.3 Work on reference planning

We have explored the relationship between planning and reference empirically and formally. Regarding the former, we have conducted a study [21] investigating referring acts and the effects of modes of communication (e.g., telephone vs teletype) on the structure of instruction-giving discourse (similar to the studies of Chapanis and his associates at Johns Hopkins [43].)

Our plan-based formalism [22] [45] that speakers would explicitly request hearers to identify the referents of their descriptions as separate steps in the dialogue. In fact, speakers frequently uttered noun phrases, alone, with questioning intonation, or as seemingly informative utterances, intending the hearer both to identify the referent of that phrase and to report on his success, as in the following fragment:

A: "Now, the red piece we talked about before?"
or
A: "There's a little red piece."
B: "Yeah."
A: "Put it in the hole on the side of that tube."

This use of language differs from the more straightforward phrasing, "Put the red piece that we talked about before in the hole on the side of that tube." We found a great difference in the use of reference strategies between telephone and teletype modes. In telephone mode, 33% of the speech acts performed, the highest percentage of any category, were requests for referent identification, compared with 10% in teletype mode. In teletype mode, requests for action were the primary means for getting the hearer to identify referents and formed the largest category (34%).

Our second line of work [46] on the problem of choosing a referring expression has been to state formal constraints on the act of referring (uttering a description with the plan that the hearer identify its referent). A condition was proposed that does not require either speaker or hearer actually to believe that the description chosen is true of its intended referent. This condition is particularly important when the system has an external link to a changing world and therefore has information that the user does not have. Under such circumstances, if system and user refer to some object O with some description D, and the world changes so that the system believes D no longer truly designates O, the system may still use D to refer to O without necessarily having to report the change.

5.2.4 The goal structure of task-oriented dialogues

The following goal structure is appropriate to all modes of our dialogues: In each such dialogue, we assume an expert and an apprentice. For each assembly goal-state (for example (CONNECT MAIN-TUBE AIR-CHAMBER), the expert has the following goals:

1. Select an action achieving that goal state.
2. Get the apprentice to perform that action.
3. Ensure success of the action.

Since we assume that the experts are creating and executing plans, goals derived from Goal (2) lead to the expert's planning and executing requests for assembly actions. In face-to-face mode, the experts achieve Goal (3) perceptually. For telephone and teletype dialogues, apprentices inform the expert when actions are completed. Occasionally, when the apprentice is taking longer than expected (or in face-to-face dialogues, when the apprentice's body is blocking the expert's view) experts question whether the requested action has been completed. These speech acts--requesting actions, reporting that actions were completed, and questioning the completion of actions--will be seen to pertain as well to the identification of referents. Before showing that the overall structuring of these two types of acts is the same, we will discuss the many ways requests are made and show that those same ways arise for identification.

5.2.5 Requests to identify

Below are various examples of requests occurring in our dialogues and a set of parallel examples of what we will call requesting identification. Requests for action will be abbreviated by "R(ACT)," and requests for identification will be labeled "R(ID)." The mode of communication is indicated in parentheses.

Direct Requests

- "Fit the blue cap over the tube end." R(ACT) (Written)
- "Notice the two side outlets on the chamber" R(ID) (Written)

Indirect Requests

- "There's a long cylinder that has a slightly purplish cast to it." R(ID) (Telephone)
- "- On the table is a small, simple red plug without a hole." R(ID) (Written)
- "do you see small/ three small red pieces?" R(ID) (Telephone)
- "Now you have two devices that that are clear plastic" R(ID) (Telephone)

These utterances show that the goal of identification can be achieved in a separate step. Because of the goal structure of our task-oriented dialogues, the success of identification requests should be questioned and reported just like requests for other kinds of acts.

5.2.6 Indirect requests to identify

In our transcripts, the problem of inferring the right intention behind an utterance form occurs frequently with identification. In the above examples, "there is . . ." is literally an informing act about the state of the world, as is "On the table is . . .," and "Now, you have two . . .". The same utterances could appear in other circumstances and would not lead a listener or reader to infer that the speaker intended an object to be identified. For example, as part of the setting of a mystery story, similar statements, whatever their function, are obviously not intended to get the readers to search their perceptual world. Our task, as analysts, is to uncover the conditions under which hearers are to make such inferences. Perrault and Allen [45] show how, by assuming hearers are trying to recognize speakers' plans, a speaker's ostensibly informing the hearer that the precondition to some act is true can be seen as requesting the hearer to perform that act, provided it is shared knowledge that the speaker would want the primary effect of that act.

Assuming that the precondition to IDENTIFY is that there exists an object perceptually accessible to the hearer satisfying the description, then, for utterances whose logical form is EXISTS (X): D(X), Perrault and Allen's general method applies. The method generates an IDENTIFY act as part of the hearer's model of the speaker's plan, provided it is shared knowledge that

the speaker has some reason to want the object to be identified (for example, to enable the hearer to perform a physical action). Thus, the plan-based inference that someone may want the precondition for an act to hold because they want to do that act generalizes to IDENTIFY.

Allen [1] suggests that properties that need to be inferred to "fill in" fragments come from shared expectations. In our domain, it is shared knowledge that the hearer will pick up pieces that the expert describes (since both know it is a manual assembly task). Given this expectation, the property in Allen's scheme that must hold is that the hearer KNOWREF D (where KNOWREF is defined as EXIST(X) [X = D] & H BELIEVE [X = D]), which is an exceedingly strong way to say that the hearer has identified the referent. Since this is the first reference to the piece in question, it is shared knowledge that the hearer has not previously identified the referent of D.

5.2.7 When are separate requests to identify used?

Major questions for computational models of language production and comprehension become:

- How and why should a system plan separate requests for identification rather than simply produce imperatives requesting assembly actions?
- Under what conditions will systems be confronted with such utterances?

Language Production: The answer to the first question is suggested by the following fragment of a teletype dialogue:

1. B: "anyway, put the red piece with the strange projections LOOSELY into the bottom hole on the main tube. Ok?"
2. A: "Which hole, the bottom one on the side?"
3. B: "right, put the 1/4 inch long 'post' into the loosely fitting hole...
4. A: I don't understand what you mean
5. B: the red piece, with the four tiny projections?
6. A: OK
7. B: just place it loosely [into the]
8. A: [done][B & A typed simultaneously, causing gibberish to appear on each screen]
9. B: yes?
10. A: yes
11. B: place it loosely into the hole on the side of the large tube...
12. A: done
13. B: very good. See the clear elbow tube?
14. A: Yes
15. B: Place the large end over that same place.
16. A: ready
17. B: take the clear dome and attach it to the end of the elbow joint...
18. A: using the blue attachment part?
19. B: right. it's already attached, so I didn't mention it Now, put the red nozzle over the hole in the dome."

There are three strategies of instruction here. First, direct requests for assembly actions, in the form of imperatives, as in line (1). Second, there are conjoined direct requests,

such as to pick up followed by an assembly action, as in (17). Finally, B performs separate requests to identify, each followed by a request for an assembly action as in (5) through (7) and (13) through (15).

It is important to notice here that B shifts his strategy (in a fashion that resembles driving a three-speed car). Prior to this fragment, the conversation had proceeded smoothly, in "high gear," so to speak, with B initially "upshifting" from first a "take and assemble" request to six consecutive assembly requests (one of them indirect), the last of which is utterance 1 of this fragment.

In (2) through (4), we observe clarification dialogue about a prior noun phrase. Immediately after an apparent breakdown at (4), B "downshifts" to questioning the achievement of his first subgoal, identifying the red piece. Once that is corrected, and a channel contention problem is solved, B stays in "low gear," explicitly ensuring success of his reference (in (13)-(14)) before requesting an assembly action in (15). After that success, he "upshifts" to "second gear" -- with requests to take and assemble in (17). After being successful yet again, B "upshifts" in (19) to "high gear," again using a direct assembly request, and stays in "high" for the rest of the dialogue (six more requests).

What could explain this conversation pattern? A representation of the plan for assembling shows clearly that to

install a piece, one must be holding it; to hold it, one must pick it up; to perform any action on an object, one must have identified that object. By requesting an assembly action ("high gear"), one requires the listener to make the remaining inferences. By requesting the sequence take-and-assemble ("second gear"), the speaker makes one of the inferences himself, but requires the listener to realize that identification of the speaker's past description is needed. Finally, "low gear" involves the speaker's checking the success of the component subgoals, which involves description identification.

In summary, the strategy shift to "low gear" occurs after a referential miscommunication, because it affords a more precise monitoring of the listener's achievement of goals.[43] A task-oriented dialogue system should have "dialogue sense" enough to plan utterances tailored to the user's problem.

5.2.8 Conclusions

A number of conclusions for natural language processing can be drawn from these results. Analyses of such utterances and their evoked responses according to a plan-based theory of speech acts indicate that, contrary to current approaches, referent identification is an action that is reasoned about in speakers'

---

[43] The use of separate utterances for identification has also been observed independently by Ochs, Schieffelin, and Pratt [42] for parent-child discourse.

and hearers' plans. Reference then becomes the act of describing something with the intention and plan that the hearer identify the referent.

In teletype mode, subjects primarily used separate requests for referent identification after prior referential miscommunication. This finding, coupled with the need to represent referent identification as an action, indicates that instruction-giving systems especially need to be able to plan referential acts as separate steps when the user is having difficulty in performing the task. To be effective, systems will have to plan their referential acts to employ the means believed to be available to the hearer for referent identification.

Finally, the difference in mode suggests that, as hearers, speech understanding systems should be designed to process these explicit requests for referent identification. Again, such systems will need to reason about identifying the referent of the user's descriptions as part of recognizing the user's plans.

5.3 A Revised Approach to One Anaphora

by B. L. Webber

5.3.1 Background

In this section, a discussion of new work on one-anaphora illustrates a somewhat different and simpler approach than that proposed in [68]. Although this approach has not been integrated

yet with the system, it appears very promising. At the conclusion of the discussion, I will comment on what is involved in its implementation.

One-anaphora, anaphoric use of the word "one" (or "ones"), is a common phenomenon in natural English discourse. Anaphoric "one"s are those that take the place of (at least) the head noun of a noun phrase. For example,

- one that I heard long ago
- the striped one you got from Harry
- three small ones.

Not all uses of "one" in English are anaphoric, of course: "one" can be used by itself as a formal, nonspecific third person pronoun - e.g.,

- One is cautioned against taunting the bears.
- One doesn't do that in polite company.

or as a numeral - e.g.,

- One true faith, two French hens, ...
- We arrived at one p.m.

Although in most cases it is easy to distinguish anaphoric from formal or numeric "one" on surface syntactic grounds alone, it is

possible for syntactically ambiguous cases to occur in text[44] - e.g.,

- Since anyone can choose his favorite number, I want one.
- Although John has bought two cats, I want one.

In linguistics, one can point to at least two significantly different approaches to one anaphora: the transformational approach (which is concerned with syntax) and the text-level approach (which is more concerned with semantics). Although the approach that I will be presenting here differs from both of these, they will be discussed briefly to provide a basis for comparison.

In transformational grammar, one-anaphora has been discussed purely syntactically, as an intra-sentence substitution phenomenon. For example, Baker [2] presents such an account in the context of deciding between two alternative structural analyses of noun phrases - the so-called "NP-S analysis" and the "Det-Nom analysis." The rewrite rules of these two analyses are roughly as follows:

NP-S	Det-Nom
NP --> NP S	NP --> Det Nom
NP --> Det N	Nom --> Nom S | Nom PP | Adj Nom
	Nom --> N

---

[44] In speech, the ambiguity may not arise because anaphoric "one" is unstressed, while the other two uses of "one" are stressed.

Baker argues for the "Det-Nom analysis" because it seems to allow the simplest statement in terms of structural identity of what "one" or "one(s)" can substitute for. Hereafter, use of "one" will always imply "ones" as well. The statement that Baker arrives at is

X	NOM	Y	ADJ the	NOM +count NUMBER	Z
1	2	3	4	5	6

condition: 2 = 5

==> 1, 2, 3, 4, one , 6
NUMBER

where a NOM inherits its features (e.g., count, NUMBER, etc.) from those of its head noun. Informally, the above transformation states that a NOM constituent preceded by an adjective or definite determiner, whose head is a count noun, can be replaced by "one" or "ones" (depending on whether the NOM is singular or plural in NUMBER) if an identical NOM appears earlier in the sentence. This transformation is meant to account for examples like

D3-1 I prefer the striped tie you got from your aunt to the paisley one.

The problem with this structural-identity account is not only that it is limited to individual sentences, but that it is not even an adequate syntactic account at that level. Consider for example the following:

D4-1 If Mary offered you a new Porsche and Sally offered you a '68 Morgan, which one would you choose?

Under no analysis does this sentence meet the structural conditions of Baker's rule: rather, "which one" means roughly "which member of the set consisting of the new Porsche Mary offered you and the '68 Morgan Sally offered you." Baker's approach has nothing to say about this.[45]

In text linguistics, a particularly clear (albeit purely informal) analysis of both definite pronoun and "one" anaphora is presented in [32]. The primary concern of the authors', Halliday and Hasan, is with the notion of "cohesion" - what makes a text hold together, what makes it more than a random set of sentences. According to the authors, both definite pronouns and "one" can provide examples of types of cohesive relations: the former, the relation of "reference," the latter, the relation of "substitution." "Reference," as Halliday and Hasan use the term, relates a text element like a definite pronoun and

> ...something else by reference to which it is interpreted in the given instance. Reference is a potentially cohesive relation because the thing that serves as the source of the interpretation may itself be an element of text. [32], pp.308-9

Except for their terminology, Halliday and Hasan's general position on definite anaphora and its relation to the discourse

---

[45] Baker poses an additional constraint on one-anaphora in [3] - effectively, a "transderivational constraint" arbitrating between optional, applicable transformational rules. However, this still treats one-anaphora purely intra-sententially and still does not address examples such as D4 above.

is not very different from that presented in [68]. That is, in processing a text, a listener or reader is building a model of the entities being discussed, their possibly changing properties, and their relationships with each other. A definite anaphor provides access - that is, it provides a handle on -- such a " discourse entity." This view is similar to that underlying Sidner's work on definite anaphora and focus [61].

"Substitution", on the other hand, is

> a formal (lexicogrammatical) relation, in which a form (word or words) is specified through the use of a grammatical signal indicating that it is to be recovered from what has gone before. The source of recovery is the text, so that the relation of substitution is basically an endophoric one. It is inherently cohesive, since it is the preceding text that provides the relevant environment in which the presupposed item is located. [32], p. 308.

Therefore, unlike definite pronouns, "one" establishes cohesion simply at the level of wording and syntactic structure. , except for its attention to more than the single sentence and for its greater concern with the <u>function</u> of "one" than with its formal syntax, Halliday and Hasan's account of "one(s)" anaphora still mirrors Baker's.

In [68], the approach to formalizing what a text makes available for one-anaphora was not too far from Halliday and Hasan's. That work is based on the view that what "one" accessed was a "description" that the speaker felt was available to the listener. Such descriptions can be made available by the

speaker's and hearer's shared spatio-temporal context, as in the case of two people peering into a geology exhibit case, one saying to the other,

"Even larger ones were found in the Mare Cambrium."

However, speakers can usually expect that descriptions they have uttered are more readily available to the listeners than this example suggests. they don't always need a shared spatio-temporal context. Hence, the most likely place to look for descriptions accessible to one-anaphora is in the text.

It is now clear that a simpler account of one-anaphora is possible. Anaphoric "one" can be viewed in much the same way as definite plural anaphors like "they," "the rocks," etc. The intuition is that "one" indicates to a listener selection from a set. That is, the interpretation of anaphoric "one" should be the same as the interpretation of "one of them". This reduces the problem to the task of identifying the set-type discourse entities (both specific and generic) that this implicit "them" can access.[46]

---

[46] Evidence for this approach also comes from Baker [2]. His rewrite rules - given above - require the "one" constituent to be interpretable as having the feature "+count" - i.e., to be capable of specifying a set. A mass term X, except when interpreted as "types of X," does not specify a set. It is also not open to one-anaphora, except in this "types of X" case - e.g.

I love red wine, especially ones that have been aged properly.

This way of treating one-anaphora may seem fairly obvious: however, its obviousness only follows from considering the sets a text makes available for access and realizing that these sets - both specific and generic - must also be "around" to provide an account of definite anaphora. As for the evidence, consider first some specific sets evoked by a text:

D5-1 All the wines in Dave's cellar are drinkable.
2 He bought them 10 years ago.

they => the wines in Dave's cellar

D6-1 Sue gave each boy a green hat.
2 She had bought them on sale.

them => the set of just mentioned green hats,
each of which Sue gave to some boy

D7-1 I see a BMW, a Porsche, and an Audi outside.
2 Do they belong to you?

they => {the just-mentioned BMW I see outside,
the just-mentioned Porsche I see outside,
the just-mentioned Audi I see outside}

(Curly brackets are used to specify a set by enumerating its members within the brackets. The symbol "=>" indicates that the definite anaphor on the left accesses the same discourse entity as the somewhat richer description on the right. The formation of such descriptions is discussed in [68], with a more succinct formulation given more recently in [69].) Each of these can be the implicit set from which a one-anaphor selects.

D8-1 All the wines in Dave's cellar are drinkable.
2 So bring me the ones he bought in Florence.

SELECT "ones" from: the wines in Dave's cellar

D9-1 Sue gave each boy a green hat.
2 Unfortunately the largest one was torn.

SELECT "one" from: the set of just-mentioned green hats, each of which Sue gave to some boy

D10-1 I see a BMW, a Porsche and an Audi outside.
2 Is one yours?

SELECT "one" from:
{the just-mentioned BMW I see outside
the just-mentioned Porsche I see outside
the just-mentioned Audi I see outside}

Notice that there may be additional stipulations in the text concerning which member or members of the anaphorically accessed set are to be selected - e.g., "bought in Florence," "largest," etc. These constraints may be used as evidence by the understanding system in figuring out what set is being accessed anaphorically. Note that these noun phrases headed by anaphoric "one" may themselves evoke new discourse entities corresponding to the selected individual or the more restricted set. Next consider some generic set-type entities evoked by a text. (These are discussed in more detail in [69]). Basically, The term "generic sets" is used for those sets characterizable as "the set of things describable as an <x>," where <x> is anything that

there could be sets of - e.g., elephants, types of beer, bottles of wine, etc.; <x> need not be any sort of "natural genus." For example, in D11 the definite pronoun "them" is not meant to access just the particular set of wines in Dave's cellar now but rather wines in general.)

D11-1 All the wines in Dave's cellar are drinkable.
2 He buys them only from the best merchants.

them => wines

D12-1 Sue gave each boy a green hat.
2 Usually she pays $8 apiece for them.

them => green hats

D13-1 I saw 7 Japanese station wagons today.
2 They must really be selling like hot cakes.

they => Japanese station wagons

Again, each of these generic sets can be the implicit set from which a one-anaphor selects.

D14-1 All the wines in Dave's cellar are drinkable.
2 So we don't need to open the ones he bought yesterday.

SELECT "ones" from: wines

D15-1 Sue gave each boy a green hat.
2 She gave Wendy one too.

SELECT "one" from: green hats

D15-3 She gave Wendy a red one.

SELECT "one" from: hats

D16-1 I saw 7 Japanese station wagons on Walnut Street and another one on Pine.

SELECT "one" from: Japanese station wagons

D16-2 They were all smaller than the French one Jean bought.

SELECT "one" from: station wagons

Notice that just as there may be more than one generic set entity derivable (via generalization) from a salient description, so may there be more than one generic set entity from which a one-anaphor may select.

To be sure, identifying what set-type discourse entities are accessible to one-anaphora cannot be done solely by the sorts of rules in [69]: subtle inferential processes seem able to make set-type entities available to one-anaphora as well. For example,

D17-1 I went to pick up Jan the other day.
2 You know, they live in that big house on Vine.

they => Jan's family

D18-1 I went to pick up Jan the other day.

2 ?? You know, the older one broke his ankle?[47]

SELECT one from: Jan's family

However, as the example shows, such entities seem less available to the implicit "of them" in a "one" anaphor. Why this is the case is not clear. Intuitively, one could say that it was easier for a listener to take an explicit "they" via Jan to Jan's family than it was to take "one" to an implicit "they" then via Jan through Jan's family to a selection from that set. The boundaries of the inferential processes that might make such associated "natural set" entities available have not been investigated. More important is the fact that as far as I can tell, no one has as yet really investigated the actual distribution of *one*-anaphora to look at the range of inferential processes assumed to be at work in such cases. Whatever the results of such investigations may be though, the unified approach to dealing with definite plural and "one" anaphora has the joint advantages of elegance and computational efficiency.

---

[47] I am aware that for some speakers it seems perfectly fine to say "You know, the one who broke his ankle?". In that case, it would seem that "one" is selecting from the generic set of of people named "Jan" that the speaker presumes the listener to know (and possibly confuse!). How proper names evoke generic sets is an object of further study.

5.3.2 Notes on implementations

Following are some notes on what is involved in augmenting a natural language understanding system for English with the ability to deal successfully with one-anaphora. I assume a RUS/PSI-KL-ONE type of frontend -- that is, a syntactically driven parser that has access to a representation of the syntactic-semantic patterns understood by the system.

The first point concerns how noun phrases (NPs) headed by "one" can be handled by a syntactic processor that wants to use such syntactic-semantic patterns for guidance and by a semantic processor that wants to use such patterns to assign (and perhaps later refine) interpretations. Clearly one does not want to have separate patterns for NPs headed by "one" and for NPs headed by "real" nouns. This would have to be the case if those patterns were only accessible through their head noun. However in PSI-KL-ONE, one identifies whether an NP matches an interpretable pattern incrementally, using information from adjectives, prepositional phrases, relative clauses, etc. as well as from the head. This identification makes it possible to interpret NPs headed by such relatively semantically empty nouns as "thing," "stuff," etc., as well as those headed by "one".

The second point concerns the processing of definite NPs headed by "one", where those NPs may themselves be anaphoric. One might infer that such NPs might have to be processed twice: once to have the anaphoric "one" resolved, then again to

determine what focused or otherwise unique discourse entity the NP accessed. However, this is not the case. Rather, simultaneously considering the constraints on the definite noun phrase raised by the anaphoricity of its head and the definiteness of the whole might simplify the problem of interpreting it. And the problem of interpretation is essentially the only thing at issue. Whether "one" is itself actually resolved or not in the process seems irrelevant.

As an extreme example, consider the following alternative sentence pairs.

D19-1 All the wines in David's cellar were given to him by friends. The ones he got from Florence are back there.

D20-1 All the wines in David's cellar were given to him by friends. The ones from Florida gave him those back there.

In the first pair, if it is assumed because of both the definite article and the one-anaphora that "the ones he got from Florence" accesses a subset of some focused set, then it should be one that can be restricted to being gotten from Florence. One can then apply this test to the focused sets to find the most reasonable one. In the second pair, the same type of assumption can be made: the definite NP accesses a subset of some focused set that can be restricted to being from Florida and being capable of giving something. In neither case need there be a separate process of resolving "one."

The third point concerns the use of syntactic information in resolving one-anaphora. If and when a "one" anaphor is to be resolved, such information can provide a basis for some useful heuristics. For example, like verb phrase ellipsis (e.g., "If Wendy runs the Bonne Bell, I will too."), one-anaphora seems to be used often in parallel constructions (e.g., "If Wendy buys a felt hat trimmed with ostrich feathers, I'll buy one with a peacock plume!"). If the system is sensitive to syntactic parallelism, information about such parallelism can be useful in doing the resolution.

The final point concerns responsibility for resolving one-anaphora. This responsibility does not belong to either syntactic or semantic processing, but rather to discourse mechanisms capable of deciding whether the "one" anaphor needs to be resolved separately or else in the process of resolving a definite noun phrase. During the resolution process discourse mechanisms will, of course, make use of information from syntactic as well as semantic processing.

6. ABSTRACT ALGORITHMS AND ARCHITECTURES

by W. A. Woods

One component of the BBN project in Knowledge Representation for Natural Language Understanding has been the exploration of specialized algorithms and architectures for the kinds of symbolic operations that would go on in intelligent language understanding systems and other knowledge based inference applications. This section presents work on specialized abstract architectures for parallel marker passing algorithms. Specifically we are concerned with discovering algorithms that can exploit high degrees of parallelism to permit real time processing of sophisticated natural language communication. One class of such algorithms involves a computerized approximation of some of Ross Quillian's [49] ideas about how human memory works. This is one of the abstract architectures that we have been considering, and the one that I will discuss here.

Quillian was concerned with how it is possible for people to put together a number of rather minor facts, no one of which is strongly suggestive, to come up with a hypothesis that integrates them into a coherent whole. He was interested in how the context of one's previous discussions can influence the way that one resolves ambiguous word senses (e.g., "tank" as a gas tank or as a military vehicle, depending upon what has previously been discussed), and he was concerned with how a person can (subconsciously and almost instantaneously) come up with a

semantic relationship between almost any two concepts. He was also interested in how the processing of such semantic associations manifests itself in human performance (for example, semantic associations between Spain and Mexico apparently affect the reaction time required to judge that sentences such as "Madrid is Mexican" are false).

Quillian's theory of such phenomena was that wavefronts of "spreading activation" passed through one's memory (a semantic network) and that "semantic intersections" of such wavefronts of activation accounted for many of the phenomena observed in human mental processing.

Quillian's work was computerized to some extent [50] but never fully developed. Scott Fahlman [24] has attempted to extend Quillian's idea in a slightly different direction - namely, if one had a machine with abilities such as those Quillian attributes to the human brain (i.e., an ability to pass waves of activation), how would one program it to perform a variety of knowledge base related inferences (such as attributing properties to instances of a class or finding all instances of a class with certain attributes). He uses a particular machine design consisting of a central machine with a large network of active memory elements; each element is able to store a small number of marker bits and respond to broadcast commands to propagate markers to other memory elements to which they are connected. In this context he has developed a number of algorithms for various kinds of inferential processing.

Fahlman's machine, however, is still not capable of many of the tasks that Quillian envisioned. For example, Quillian's account of how one's context influenced his subsequent interpretation of ambiguous words assumed that "activation" from prior discussion remained associated with a Concept for some time after its discussion and that a choice between two Concepts underlying an ambiguous word would be made in favor of the Concept with the higher residual activation. The Fahlman machine, however, has only a small number of marks, which must be constantly recycled; therefore, it would not be able to account for such behavior using marker bits as the traces of activation.

Moreover, the Fahlman machine, while permitting considerable parallel processing in the propagation of markers, still requires the central processor to be devoted to a single function at a time, coordinating all of the marker propagation to a single purpose. He cannot independently pursue two different hypotheses at the same time. Quillian's vision, or at least my extrapolation of it, would permit marker passing processes to be working on separate parts of an overall solution to a problem, or even alternative solutions to a problem simultaneously. This kind of processing requires a much larger number of marks, in fact a number that is effectively open-ended (in somewhat the same way that a computer's finite memory is considered open-ended, although it is finite in actuality).

I have been exploring a class of parallel algorithms that is

somewhat similar to Fahlman's, except that the number of marks is large - in the thousands or greater - and can be used to coordinate the activities of a large number of alternative inferential operations that can be pursued in parallel. I visualize these algorithms being used in operations such as recognizing a scene, parsing a sentence, recognizing the intent of an utterance, recognizing a standard situation in a planning task, recognizing an instance of something of importance (as in an alerting system), and recognizing instances of counterevidence for hypotheses in learning systems. All of these applications and many other similar ones that occur throughout applications of artificial intelligence, have the characteristic that potentially many rules, patterns, templates, or whatever need to be matched against the situation, so that the better matching ones can be discovered. Moreover, in general, the situation involves not just the discovery of one instantiated pattern, but rather the "parsing" of the situation into a constellation of related structures according to known rules of composition, just as sentences are parsed into overall structures of related phrases. I refer to such processes as "high-level perception" or "situation recognition."

6.1 Parallel Algorithms for High-Level Perception

Most successful parallel processing architectures have resulted from designing an architecture to support some specific class of algorithm that is known to be important for some useful

class of problems, but computationally expensive. Examples are Fast Fourier Transforms (for signal processing applications) and Key retrieval for associative memory processors. The former has turned out to be immensely important for a variety of signal processing applications, while the latter turns out to be extremely useful for storage management in virtual memory systems and presumably has many other useful applications as well.

However, contrary to what one might have expected, the intuitive attractiveness of an associative memory for various kinds of Artificial Intelligence applications and other sophisticated symbolic processing applications has not resulted in the widespread use of associative processors for these applications. The reason for this, I believe, is that the kinds of retrievals required for most sophisticated applications are not merely direct retrieval of a known key, but rather the discovery of a stored item that matches a complex pattern. Some associative processors permit patterns that can be expressed as "don't care conditions" on certain bit positions in the key, but this is not nearly sufficient.

I believe that the problem of high level perception or situation recognition is sufficiently central to a wide range of artificial intelligence applications to make it appropriate for special architectural treatment. Moreover, it appears that some version of Quillian's spreading activation ideas may provide a method for obtaining real-time performance in such tasks.

Appropriate algorithms and specialized hardware for such operations hold promise for significant improvements in elapsed real time required to perform complex intelligent computations in knowledge-based systems.

The problem of high level perception or situation recognition is to find all of the patterns in a data base of pattern schema that are satisfied by a given input, where the class of possible patterns is as general as possible. This basic recognition problem is a fundamental "inner loop" operation in almost all sophisticated "intelligent" symbolic processing applications, including language understanding, visual perception, medical diagnosis, mechanical inference, robot problem-solving, automatic program synthesis, and the general class of knowledge-based systems. The efficiency of this basic process is critical to any rule based system that is to operate with a large and sophisticated set of rules.

6.2 Situation Recognition Algorithms

As discussed above, in a system whose behavior is driven by a large set of rules, a major inner-loop operation is the determination of the set of rules that are applicable to a given situation. For sufficiently small sets of rules, this can be done by considering each rule in turn and matching its conditions against the current situation (e.g., matching a production rule against the contents of a specified set of registers). For larger sets of rules, this approach becomes impractical.

For a straightforward set of production rules, an efficient means of finding matching rules is to use a decision tree made up of the various tests that occur in the patterns of the production rules, and to store a list of the corresponding rules at each of the leaves of this tree. This method significantly decreases the computation for determining the matching rules.

As the number and complexity of the rules increase, the decision tree method faces two alternative strategies. Either the decision tree can be interpreted as a deterministic automaton (in which only a single path through the tree is followed), or it can be interpreted as a "nondeterministic" automaton (in which alternative paths through the tree can be explored simultaneously).

Woods [75] presents an analysis of these options in which it is clear that the deterministic decision tree is impractical. It also points out that the nondeterministic decision tree can be viewed as a special case of an ATN, and that some generalizations of the notion of nondeterministic decision tree in the directions of an ATN have a number of advantages for organizing systems of production rules. Of the three options (separate matching of each rule, deterministic decision trees, and nondeterministic decision trees), the nondeterministic decision tree seems clearly superior to the other two. It involves fewer elementary measurements than the straightforward testing of each rule in turn, and in some cases could reduce the number of evaluations

from order n to order log(n). Moreover, the storage costs are no greater and could be less than that of the direct representation of each rule. Both the direct evaluation of each rule and the nondeterministic decision tree permit vast amounts of potential parallelism for systems with large numbers of rules.

The observation that the nondeterministic decision tree is essentially an ATN with no pushing and no registers raises the possibility that one might be able to use more complex rule patterns, or have greater flexibility for compact expression of a total collection of rules, by generalizing to unrestricted ATN's and making use of pushing and register setting. This is in fact the case. Conventional production rule systems assume that the pattern parts of the rules are essentially "flat" Boolean combinations of simple measurements on the current state. They do not provide a significant capability to create the equivalent of subroutines that operate on some subset of the situational parameters and may be common to a number of different rules. By the use of factoring, pushing, and a device analogous to well-formed substring tables, it is possible to make constellations of tests into subroutines that need only be executed once, even though they may be used on several different paths through the nondeterministic network. The use of registers will then permit a computation path to keep a record of the results of such subroutines. The use of registers can also permit the merger of two similar parts of the network into a single structure, with the differences between the two kept in the registers.

The above analogy to ATN grammars may seem strange to those accustomed to thinking of grammars only as devices for parsing strings of symbols. However, the ability to do arbitrary tests on an arc permits an ATN to be used essentially as a universal nondeterministic computing machine. This ability has been partially formalized in Woods [78] into a generalization of ATN grammars called a "Generalized Transition Network," or GTN. A GTN permits the same kind of nondeterministic computation as an ATN, but is generalized to analyzing an arbitrary "perceptual domain." In a GTN, one characterizes the situations that can be induced from a set of rules, and then associates the right-hand sides of the rules with the situations in which they should be executed. Viewed in terms of GTN's, the problem of determining which rules to apply to a given situation is equivalent to parsing that situation into a characterization, which then determines which rule actions are applicable.

Once this perspective is taken, it becomes clear that the use of a GTN to characterize situations also permits a more extensive classification of possible interactions among sets of simultaneously matching rules than merely a set of matching rules. For example, if one rule has a more specific pattern than that of another rule, then whenever the first rule matches, the second rule will also. If the rules are taken as merely a set of rules, then these two rules will compete with each other in the cases where the more specific rule matches. In the GTN formulation, it is possible to indicate that in certain

situations one rule is to take precedence over another, while in others the effects of the two rules are to be combined, and in still others the rules should be considered as competing alternatives.

A previous report [78] discusses motivations for introducing mechanisms of inheritance into the specification of a GTN for situation-recognition applications. A structured inheritance network such as KL-ONE [10, 11] can be interpreted as specifying a GTN that provides such an inheritance mechanism and recognizes occurrences of the concepts described in the network. That is, a given Concept with a particular set of Roles and structural conditions can be thought of as having an associated GTN (not necessarily explicitly constructed) whose states are characterized by subsets of the Roles that have been filled and whose transitions correspond to filling an additional Role. Such a network can be used as a "taxonomic lattice" [74] to organize "advice" to be acted upon in different situations. Such a system has a number of advantages over conventional production rule systems, as I have discussed above.

One can imagine a process that simultaneously attempts to parse an input into all of the possible Concepts that are known to the network. However, what one would really like to find are the most specific Concepts in the inheritance network that can parse the input (the more general Concepts will then be inherited automatically). Thus an important algorithm for situation

recognition is an algorithm that takes a given input situation and finds the most specific Concepts in the system's taxonomy that are satisfied by the input situation. It is algorithms for such operations as this that we are searching for.

6.3 An Abstract Marker Passing Engine

The abstract architecture that I have been exploring consists of a central controller and a collection of processing nodes, each of which contains a certain amount of permanent and scratch memory. Each node is connected to some number of neighbors by a communication channel called a link. A node can pass a message to another node along the link without contention with other communication links, subject only to the contention of several nodes simultaneously trying to send messages to a single destination. The central controller can broadcast messages to all nodes, and nodes can send messages to the controller through a contention network that resolves competition for the controller's attention according to priorities associated with the messages.

This architecture is superficially similar to that of Fahlman, although we hypothesize a more powerful node than he does, as well as a greater ability for parallel, asynchronous activity without the detailed sequential supervision of the central controller. We will use the capabilities of the controller to regulate and modify the activities of the nodes and to make "policy" decisions on courses of action.

We assume that the nodes of such a machine are used to implement a taxonomic network, as in KL-ONE, with an individual processor node for each of the KL-ONE Concepts, Roles, and Structural Descriptions (this is for the abstract machine architecture - a real implementation might use a single processor to simulate some number of these abstract processing nodes). Links of the architecture will correspond to the various kinds of KL-ONE links. One can think of the architecture, then, as a KL-ONE network with a rudimentary processing ability at each of its nodes and an ability to send messages along its links, plus the ability for the nodes to communicate with the central controller.

The communication between nodes in this architecture will take the form of messages of a particularly constrained format, somewhat comparable to that of a machine instruction for a computer with a two- or three-address instruction format. In the current conception, a message will consist of a finite state propagation code, a priority number, and one or two fields containing bit patterns called "identifiers." The propagation code will be used to govern the kind of operation that will happen when a node receives a message (e.g., there can be a propagation code for following SuperConcept (Superc) links and marking nodes along such a chain). The identifiers are bit patterns assigned by the central controller to different tasks and are used to "color" different waves of activation that are simultaneously advancing through the network so that they can be kept separated and associated with the tasks that they are

performing. Marking a node will consist of storing a marker at the node consisting of a finite state status code, one or two identifiers, and a priority code or "activation strength."

The priority code associated with a mark on a node can serve several purposes depending on the status code. These include:

- to affect the priority of messages generated by collisions of other markers with this one
- as a time constant for an age counter that keeps track of how long a mark has been present on a node and then forgets it after a specified period
- as a measure of the goodness of a match between the criteria for some concept and the external evidence being used to hypothesize an instance of it
- as a regulator of the speed with which a wave of markers propagates through the network by specifying a number of cycles that a message should wait after being sent to a node before having its effect

The full range of uses of the activation strength parameter has yet to be worked out. It may eventually be necessary to hypothesize several distinct priority fields for simultaneous specification of several of the above behaviors.

We will make the following assumptions:

1. The system will contain a finite number of identifiers (probably numbering a thousand or more) that can be associated with the nodes of the network in a number of different ways (indicated by a status code in a marker stored at the node).
2. A node may hold a number of markers simultaneously.
3. Markers can be propagated through the network by means of messages whose propagation code governs the route of the propagation, the kind of markers that will be

stored at nodes, the detection of "intersections" with other markers stored at nodes (a la Quillian's "semantic intersections"), and the generation of messages as a result of such intersections.

4. A node can request an identifier from a list of available ones managed by the central controller.[48]

5. A node can send a message to the central controller.

6. The central controller can broadcast a message to all nodes of a given type.[49]

7. A node can count cycles after a message has been received and execute the effect of a message after a specified time delay.

In place of addresses, this machine will use the identifiers discussed in assumption 1 above, which will be considerably fewer in number than the number of nodes in the memory structure. Identifiers can be reused for different purposes at different stages of a computation, just as registers and temporary variables are used in conventional programming. However, instead of storing data or pointers to data in registers in a central processing unit, markers will be attached to the data in the

---

[48] This may require freeing up and recycling an old identifier - garbage collection is an open topic in this architecture.

[49] Each type could have a separate broadcast channel, or each node could have an automatic detector that ignores broadcast messages whose type is different from that of the node. The system would have a small number of types (say from 1 to 6). In addition to setting and clearing markers, this channel might be used to reprogram the nodes by specifying a program in some microprogramming language that defines the effects of the different propagation codes.

memory nodes themselves, indicating the status of the data and the task for which it has that status.

For example, if a marker is put on a node with a status called OWN, its identifier can be used as a temporary handle on the node so marked - i.e., the node could respond to broadcast requests for the owner of that identifier. Coupled with the ability to respond to broadcast requests, this use of markers provides a way of implementing a selective addressing scheme, which obviates the need for a node to have an address decodable as a pointer in order to access it. For large semantic networks, fewer bits will be required for such markers than would be required for equivalent pointers, and this in turn will reduce the number of bits of communication required among nodes for various kinds of algorithms.

This use of markers as node handles to hold onto intermediate results, as opposed to using temporary registers or variables to store pointers to addresses, can not only reduce the bit rate of communication, it can potentially remove some contention from communication channels. Instructions to mark nodes can be broadcast without requiring that pointers to the marked nodes be returned to registers in the central processor. This use of markers also has the advantage of permitting the equivalent of multiple (nondeterministic) setting of registers by letting a given marker be assigned to several nodes. Subsequent broadcast instructions to perform operations on the node(s) so

marked will automatically be performed in parallel for all possible values of this "nondeterministic variable."

The use of markers instead of registers thus has a number of advantages for realizing the potential parallelism of high level perception processes:

1. It reduces the communication bottleneck that occurs in a memory access when a pointer is transferred from an arbitrary point in memory into a central register. An operation to move the "contents" of one marker to another consists of merely attaching an additional marker to those contents, without the necessity of placing a pointer to those contents into a central register and then storing the contents of that register in another place).

2. It permits the simultaneous assignment of different values to the same marker and the parallel computation on the different hypotheses that correspond to those different values. For example, if one wants to determine if any node with some condition A also has condition B, one can arrange all nodes with condition A to be marked with some marker m, and then simultaneously launch the computation for condition B for all nodes marked m.

3. Finally, since the number of identifiers is smaller than the number of nodes, the messages being passed in such processes can be smaller than the corresponding messages using pointers would be.

Another use of markers is to construct temporary connections between different nodes in the network. For example, one can take the simultaneous presence of a marker on node A with identifier m and status SOURCE and a marker on node B with the same identifier and status DEST as a virtual link from A to B. If one wants to assign such a link specific properties, one can associate some node M with that link by assigning it a marker

with the same identifier m and status OWN, and then associate those properties with that node. Such virtual links make it possible to build up short-term memory representations of potential network structure that may or may not be eventually incorporated into long-term memory. Such short-term "scratch" structures can be used in hypothesizing alternative interpretations of input sentences, planning future actions, making internal inference steps, etc. If nothing is done to make such virtual links permanent, they can be made to disappear when their marks become sufficiently old[50] and are collected.

Pairs of identifiers can be associated with nodes to indicate some connection between two tasks or to detect and respond to "intersections." For example, a marker with a pair of identifiers m1 and m2 and status R,S-TRANS (referred to as a "TRANS" pair) can be used to indicate that if a node receives a marker with identifier m1 and status R, it should send to itself[51] a marker with identifier m2 and status S. Such a TRANS marker can be left on a node by one spreading activation wavefront and activated by the appropriate marker from a second

---

50 Markers can be aged in at least two ways - either by the central controller or by including self-destruct clocks with the markers at the nodes.

51 The notion of a node sending a message to itself is distinct from the notion of storing a message in that it will cause the testing of all of the same propagation rules as if the message were received from somewhere else. It need not actually tie up the communication medium used to connect nodes together, however.

wavefront to cause a new wave of propagation to begin from the marker created by the interaction of the two.

Many applications of marker passing will be for problems of search through a network. In such cases, there may be priorities associated with alternative hypotheses, and for this reason one may want to associate priorities with messages being passed along the links in the network. For example, such priorities might be used to score the relative likelihoods of different possible semantic interpretations of a sentence. The priorities could then govern the access of the individual marker propagations to bottleneck resources such as requests from the central controller. The priorities could also affect the passage of marks through congested regions of the network by giving precedence for service at a node to higher priority markers. The use of priorities associated with a marker could thus affect the speed of propagation through the network and serve automatically to adjudicate any competition for machine resources among competing hypotheses in a search. As another example, higher priorities could be used on certain kinds of "cancel" markers in order to catch up with and terminate computations that have already been initiated but are subsequently discovered to be unnecessary, because of the success of some other subcomputation.

To illustrate more concretely the kinds of virtual structures that can be built with markers, suppose that as a possible interpretation of a part of an input utterance, we

wanted to represent the proposition that the ship Enterprise is located in Oceana, Virginia. Let us suppose that Concept nodes contain enough internal memory to store n-tuples of identifiers, with some association (either explicit or implicit) between the positions in the n-tuple and the Roles of the Concept. These n-tuples can be used to store temporary individuators of the Concept involved. The first component of the n-tuple, corresponding to the "whole" Role that stands for an instance of the Concept as a whole will contain a unique identifier to be used as a handle on the temporary individuator being constructed. The remaining identifiers will be the identifiers for the Concepts that are to be used as the fillers for the respective Roles. Then, assuming that the Concepts [Enterprise] and [Oceana] have already been marked as owners of the identifiers m1 and m2, respectively, we can represent the Enterprise being in Oceana by an n-tuple (m3 m1 m2) stored under the Concept [a ship being located at a place], where m1 is associated with the [ship] Role of the Concept, and m2 is associated with the [place] Role. The identifier m3 can now be used as a handle on the hypothesized individuator [Enterprise being located in Oceana]. If it is later decided to make this information a permanent part of long term memory, a process can be invoked to create a new individuator node whose SuperConcept is the [a ship being located at a place] node, and whose [ship] Role is filled with [Enterprise], and whose [place] Role is filled with [Oceana]. The configuration of markers discussed here is shown in Figure 14.

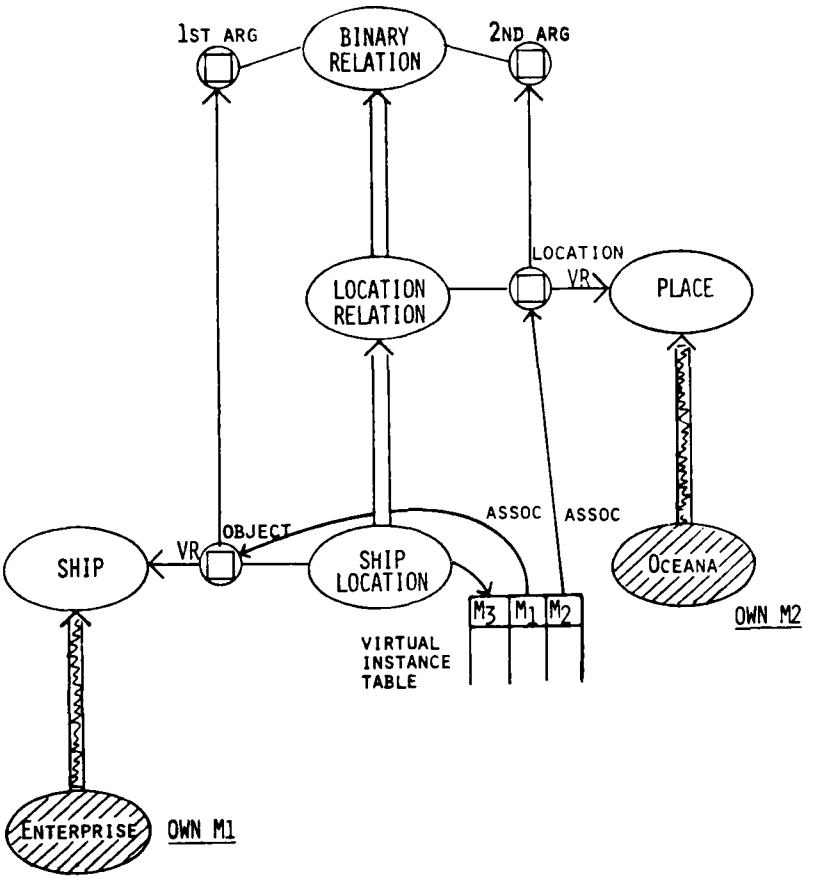


FIG. 14. "THE ENTERPRISE IS LOCATED IN OCEANA." VIRTUAL INSTANCE REPRESENTED BY VECTOR TABLE.

A possible way of implementing the n-tuples with correspondences to Roles would be to use nodes in the network that are exactly like individuator nodes, but without any Roles filled in. The virtual filling of these Roles could be done with a marker that specified the identifier of the Role filler. Each generic Concept could have a certain number of such nodes, reflecting the number of individuators of that Concept that might

have to be independently hypothesized at one time. Additional such individuators could be created whenever the number of them was found to be insufficient. When a permanent individuator was to be created, the temporary markers associated with the Roles of one of these special nodes could then be replaced with actual Val links to the Concepts that owned their identifiers, and a new empty individuator of the generic Concept could be created (either immediately, or later, when needed) to replace the one that had just been used up. The configuration of markers for the location of the Enterprise example using this convention is shown in Figure 15.

Another possible realization of this kind of virtual instance is to dispense with the explicit storage of n-tuples and make the association of fillers with Roles by placing a marker that pairs the virtual instance identifier and the Role filler identifier on the Role being filled. A marker with status code IND, associated with the generic Concept that the virtual instance individuates, would be sufficient to indicate the presence of a virtual instance and specify its identifier. That identifier then could be used to find its Role fillers by looking up the network for the Roles that Concept inherits and finding the Role filler information at those Roles. The configuration of markers for the location of the Enterprise example in this convention is shown in Figure 16.

Figure 17 shows in dotted lines the virtual structure that this configuration of markers encodes.

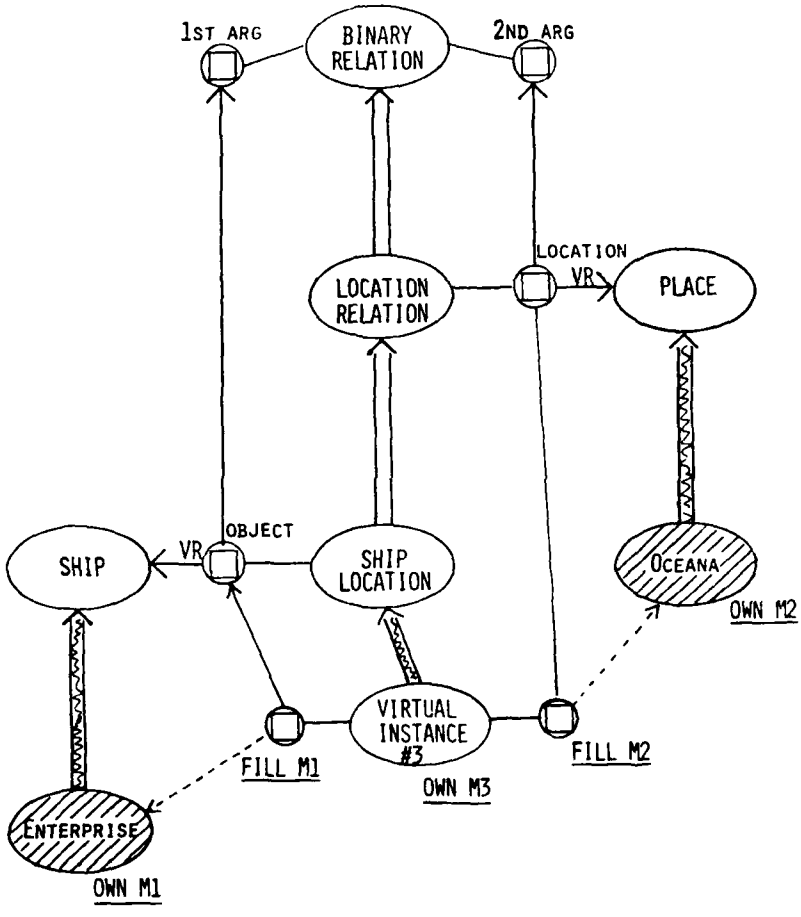


FIG. 15. "THE ENTERPRISE IS LOCATED IN OCEANA." VIRTUAL INSTANCE REPRESENTED BY SPECIAL INDIVIDUAL CONCEPTS.

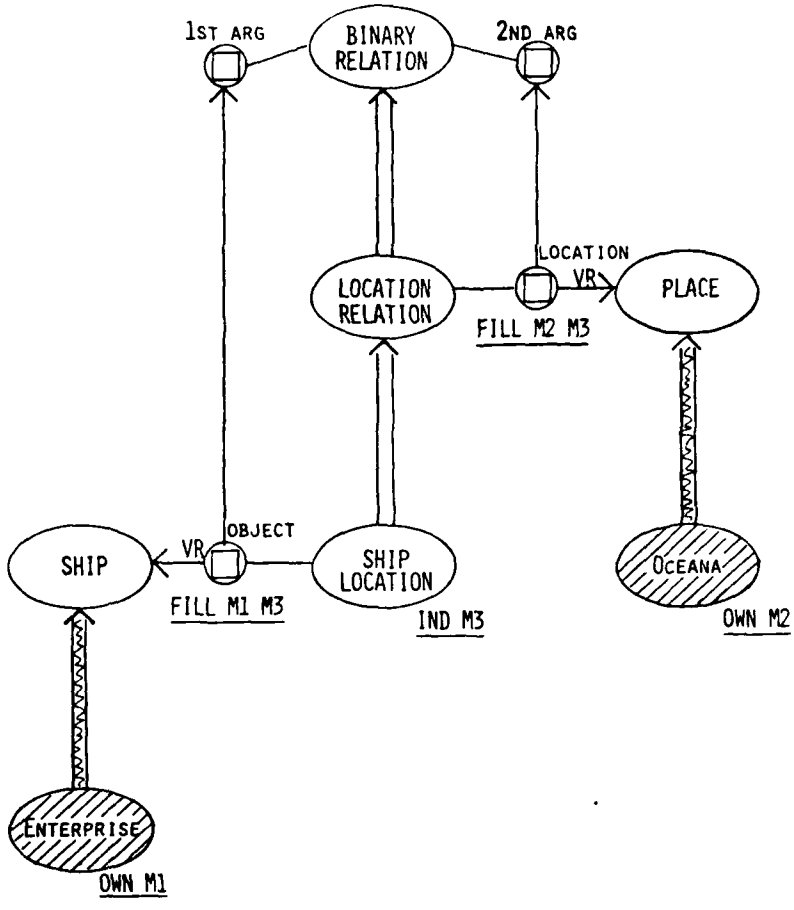


FIG. 16. "THE ENTERPRISE IS LOCATED IN OCEANA." VIRTUAL INSTANCE REPRESENTED BY IDENTIFIER/FILLER PAIRINGS ON ROLES.

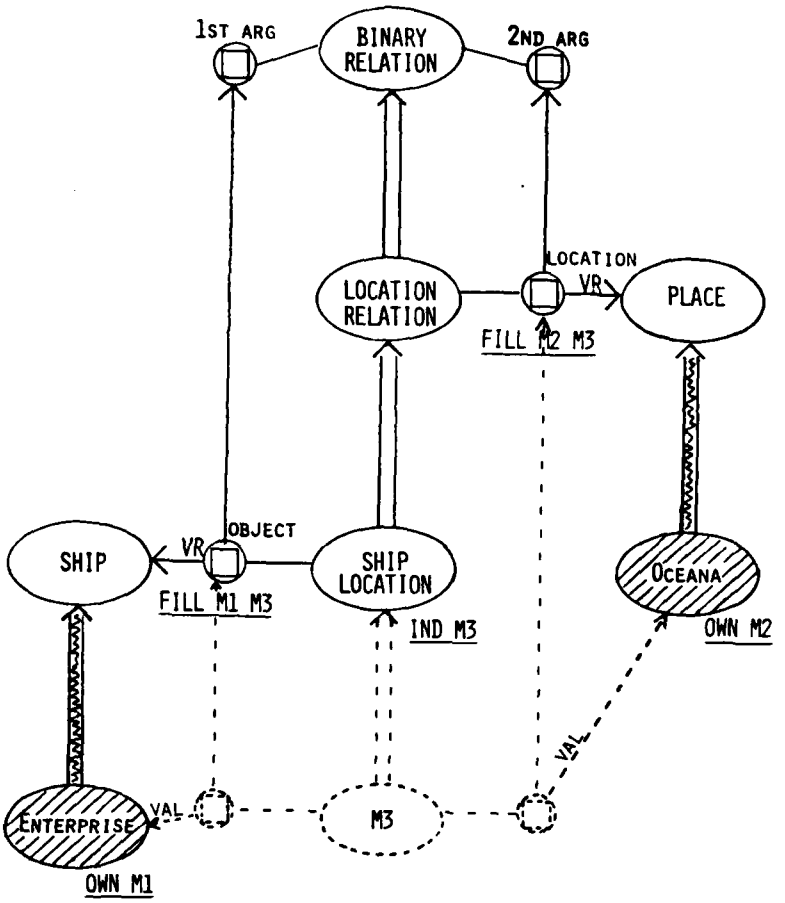


FIG. 17. "THE ENTERPRISE IS LOCATED IN OCEANA." SHOWING VIRTUAL INSTANCE INDUCED BY MARKERS.

6.4 Specification of Marker-Passing Algorithms

A marker-passing algorithm is expressed by determining a set of status and propagation codes to use for various purposes and a set of propagation rules for passing markers through the network. An algorithm is then evoked by marking certain nodes in the network with markers corresponding to its arguments and waiting for the consequences of those initial markers to propagate through the net. The algorithm "returns" its results by leaving certain nodes marked with a status indicating that they are the results. In some cases, the central controller may then access these results by broadcasting a request that they respond, but in many cases, the mere fact that they have been marked will constitute the input for another wave of marker propagation to compute some other inference.

In actual practice on a real implementation, the propagation and status codes would consist of bit patterns that would serve to access the corresponding propagation rules, which themselves would be specified in something like a machine instruction format. However, for specification purposes by a programmer, we would like a more readable surface notation. At this level, a programmer will specify mnemonic names for the propagation and status codes that he intends to use and will specify the propagation rules by means of a simple pattern-action notation. A sample propagation rule could be specified by a surface notation something like:

```
(WHEN A CONCEPT GETS SUB M
      AND DOESN'T HAVE SUB M :
      STORE SUB M,
      SEND TO SUPERC : SUB M),
```

which would specify that a message with propagation code SUB and identifier M, when received by a node that did not already have a marker with status SUB and the same identifier, would cause the storage of a marker SUB M at that node and the sending of the message SUB M to all SuperConcepts of that node. Such a rule would be compiled into an appropriate microlanguage and stored at each node in the network (or otherwise made available to each node), or it could even be "wired" into the network nodes directly in a machine whose inventory of propagation and status codes was fixed.

6.5 The MSS Algorithm - An Example

To illustrate the use of marker passing algorithms to exploit the potential for parallelism in knowledge based inference, I will give here an example of a marker passing implementation of the MSS algorithm, used to find the most specific concepts in a KL-ONE network that subsume a given input description. This algorithm is one of the basic inferential operations on a KL-ONE network. It is used in assimilating new Concepts into the network to determine what Concepts the new Concept should inherit information from. The most specific subsumers of a Concept are the Concepts to which it should be linked by SuperConcept links.

Assume that Concepts can be described by list structured expressions characterized by the following simple BNF grammar:

```
<cspec> -> ( <cname> <rspec>* )
<rspec> -> ( <rname> <cspec>* ),
```

where "cspec" stands for a Concept specification and "rspec" stands for a Role specification. In this notation, a Concept is specified by a Concept name (cname) designating a Concept in the network under which the specified Concept will fit, and a collection of Role specifications that the specified Concept must have. Similarly, a Role specification will consist of the name of a parent role, plus Concept specifications for value restrictions that the specified Role must have. (This is a simplification of the range of information that is actually contained in a KL-ONE Concept, but the simplification makes for a much more comprehensible example.) An example of a Concept specification in this notation is:

```
(PERSON (HOBBY (GOLF)) (OCCUPATION (POLITICS)))
```

corresponding to the description "a person whose hobby is golf and whose occupation is politics."

A Concept in a KL-ONE network is said to subsume such a description if anything satisfying the description would be an instance of that Concept. For example, the above description would be subsumed by the Concept PERSON, and by Concepts for people whose hobby is golf, people whose hobby is golf and whose

occupation is politics, people whose hobbies include golf, people whose hobbies are outdoor sports, etc. The subsumption relationship between KL-ONE Concepts and descriptions can be defined recursively by the conditions:

1. A Concept subsumes a set of cspecs (i.e., subsumes the conjunction of those cspecs) if either (a) its name is the same as the cname of one of those cspecs, (b) it lies on a SuperConcept chain above a Concept that satisfies condition a, or (c) all of its immediate SuperConcepts subsume the set of cspecs and each of its attached Roles subsumes some rspec from some one of the cspecs.

2. A Role subsumes an rspec if it has or inherits the name of the rspec and each of its value restrictions subsumes the set of cspecs of the rspec.

This recursive characterization of subsumption can be used as the basis of a strategy for assigning a given marker to all Concepts in a network that subsume a given description. From there it is a simple additional step to propagate a "cancellation" marker up SuperConcept chains from each such Concept, so that only the most specific subsumers have the subsumption marker and not the cancellation marker.

Recalling the basic structure of a marker passing algorithm, we need first to select a set of status and propagation codes to serve various purposes and then set up a set of propagation rules for passing markers through the network. For our example MSS algorithm, we will make use of the following status codes:

SUB a status used to mark a Concept that subsumes a cspec or a role that subsumes an rspec. The MSS algorithm, when given a cspec and an identifier m

as arguments, will arrange to assign a marker with identifier m and status SUB to all Concepts that subsume that cspec.

TEST — a status used to mark a Concept that is a possible subsumer of a cspec, depending on whether its other SuperConcepts and its roles satisfy the recursive conditions for subsumption.

FOUND — a status that will be used to mark a Role that has been determined to satisfy the recursive subsumption conditions.

TRANS — a status for a pair of identifiers indicating that when a marker with the first identifier of the pair is received, a marker with the second identifier of the pair should be sent or stored.

The overall strategy of our MSS algorithm will be for the central controller to start with an identifier cm assigned to the particular cspec being processed (there may be any number of simultaneous MSS operations being carried out at any given time, and this identifier will be used to keep the separate computations from mixing up their intermediate results). A message consisting of the identifier cm with propagation code SUB will then be broadcast by the central controller to the concept named by the cname of the cspec. (This message will cause a marker SUB,cm to be stored at that node, a message with the same format to propagate up SuperConcept chains from that node, and a marker with propagation code TEST and the same identifier to propagate down SuperConcept chains.) An identifier rm will then be chosen by the central controller (i.e., a new identifier is requested from the central controller) for each of the rspecs of the cspec. A message TRANS,rm,cm will be broadcast to all roles in the network with the rname of that rspec. This broadcast sets

up a condition that marks each such role as subsuming with respect to cm if all of its VR's send a message FOUND,rm (indicating that all the value restrictions of that role subsume cspecs in the rspec associated with rm). Finally, for each of the cspecs of the rspec associated with rm, MSS will be called recursively on that cspec using the identifier rm, thus setting up conditions so that all Concepts that subsume the cspecs of this rspec will eventually be marked SUB,rm and will send a message FOUND,rm to their inverse VR's (i.e., the roles of which they are value restrictions).

An algorithm for injecting the necessary markers into the network can be written as follows:

```
(DEFINE (MSS CM CSPEC)
    (BROADCAST TO CONCEPTS WHOSE NAME =
               (CNAME CSPEC) :  SUB CM)
    (FOR RSPEC IN (RSPECS CSPEC) DO
         RM <- (GETMARK)
         (BROADCAST TO ROLES WHOSE NAME =
                    (RNAME RSPEC) :
                    TRANS RM CM)
         (FOR CS IN (CSPECS RSPEC) DO
              (MSS RM CS) )))
```

where MSS is the function that initiates the MSS algorithm with identifier CM for description CSPEC, BROADCAST is a function that broadcasts messages to Roles and Concepts (possibly restricted to have a specified name), and GETMARK is a function that requests a new identifier from the controller.

In order for the signals injected by the above MSS function to have their effect, appropriate propagation rules must be set

up to pass markers along the links of the network, changing status as appropriate, and interacting with other markers as indicated by trans pairs. A set of marker propagation rules for our example MSS algorithm is given below.

```
(WHEN A CONCEPT GETS SUB M
      AND DOESN'T HAVE SUB M :
      STORE SUB M,
      SEND TO SUPERC : SUB M,
      SEND TO SUBC : TEST M,
      SEND TO INVERSE VR : FOUND M)

(WHEN A NON MAGIC CONCEPT GETS FOUND M,
      AND ALL OF ITS ROLE HAVE SUB M,
      AND ALL OF ITS SUPERC HAVE SUB M :
      SEND TO SELF : SUB M)

(WHEN A NON MAGIC CONCEPT GETS TEST M, AND DOESN'T HAVE SUB M
      AND ALL OF ITS ROLE HAVE SUB M,
      AND ALL OF ITS SUPERC HAVE SUB M :
      SEND TO SELF : SUB M)

(WHEN A ROLE GETS TRANS M1 M2,
      AND DOESN'T HAVE TRANS M1 M2 :
      STORE TRANS M1 M2,
      SEND TO SUBROLE : TRANS M1 M2)

(WHEN A ROLE GETS FOUND M1,
      AND HAS TRANS M1 M2,
      AND ALL ITS VR HAVE SUB M1 :
      STORE SUB M2,
      SEND TO ITS CONCEPT : FOUND M2)

(WHEN A ROLE GETS TRANS M1 M2,
      AND ALL ITS VR HAVE SUB M1 :
      STORE SUB M2,
      SEND TO ITS CONCEPT : FOUND M2)
```

This set of propagation rules, together with the above MSS function for injecting markers into the network, will cause all those Concepts in the network which subsume a given cspec to be marked with the identifier assigned to that cspec in status SUB.

The restriction to non-MAGIC[52] Concepts in two of the clauses above is necessary to block the apparent subsumption of a cspec by Concepts whose definition is not totally indicated by its Roles and SuperConcepts. The placement of a new Concept under a MAGIC Concept cannot be done by an automatic MSS algorithm, since some additional source (the "magic") must be consulted to determine whether the candidate Concept is really subsumed by such a Concept. For example, a Concept for "mouse", characterized in the network only as a mammal whose size is small and color is gray, but associated with a recognition procedure that could determine from a visual shape description whether an animal was a mouse, should not automatically subsume any small gray mammal. Concepts in KL-ONE can be marked as MAGIC to indicate their dependence on such additional criteria. In this case, the procedural shape recognizer would constitute the additional (MAGIC) source of information necessary to determine whether a Concept could be subsumed under the mouse concept and would be the only procedure authorized to put a SUB marker on that Concept. The placement of such a marker by this procedure

---

[52] The notion of MAGIC concepts is used in KL-ONE as an escape mechanism to indicate that some knowledge external to the knowledge representation system is required to determine whether an item should be classified under a given concept. Such knowledge might be embedded in the operation of some low level perceptual routine that makes assertions into the knowledge base, or may reside in some external agent that must be consulted before making such a classification. MAGIC concepts are somewhat similar to the philosopher's notion of natural kind terms.

could begin waves of marker passing, but no other marker passing wavefronts should be allowed to place a SUB marker there without having verified the procedural conditions of the mouse recognizer. This does not preclude other marker passing algorithms than MSS from placing various markers on this Concept (e.g., the mouse recognizer itself might be implemented as a marker passing algorithm).

Notice that any number of simultaneous MSS computations can be going on in the network at any given moment, each with its own identifier to keep it distinct from the others. In fact, the above MSS algorithm creates this kind of simultaneous activity in carrying out the MSS subsumption on the value restrictions of its Roles.

The elapsed real time required by this algorithm to compute a MSS is proportional to the longest chain of marker propagation necessary to complete the computation, which is in general much shorter than the number of machine instructions that would be required to compute the MSS operation on a sequential machine. This necessary elapsed time depends on the depth of the taxonomic lattice rather than on its breadth or on the total number of Concepts. Moreover, the depth of the lattice will tend to be on the order of log(n) for a lattice of n Concepts, so one would expect a real time requirement that grows on the order of the log of the number of Concepts in the network. (The time required for MSS to inject markers into the network is negligible since it is

bounded by the number of elements in the input cspec, which is in general quite small.)

6.6 A Trace of the Example

To illustrate the operation of the above example of the MSS algorithm, consider the situation shown in Figure 18, which shows a simple network for the concepts man, sports person, sport, and golf, together with an initial marker assignment corresponding to an invocation of the MSS algorithm for "a man whose hobby is golf." (Asterisks mark the magic Concepts in the figure - i.e., the information that distinguishes a person from an arbitrary thing is not shown in the network, nor is the information that distinguishes a man from a person.) Figure 19 shows a history of the marker passing that results from broadcasting the messages SUB,M1 to the Concept MAN, TRANS,M2,M1 to Roles named HOBBY, and SUB,M2 to the Concept GOLF. Figure 20 shows the resulting pattern of stored markers when the processing ceases. To get from here, which is really just a marking of all Concepts that subsume the target, to a marking of just those Concepts that are most specific subsumers, would require some additional propagation rules such as:

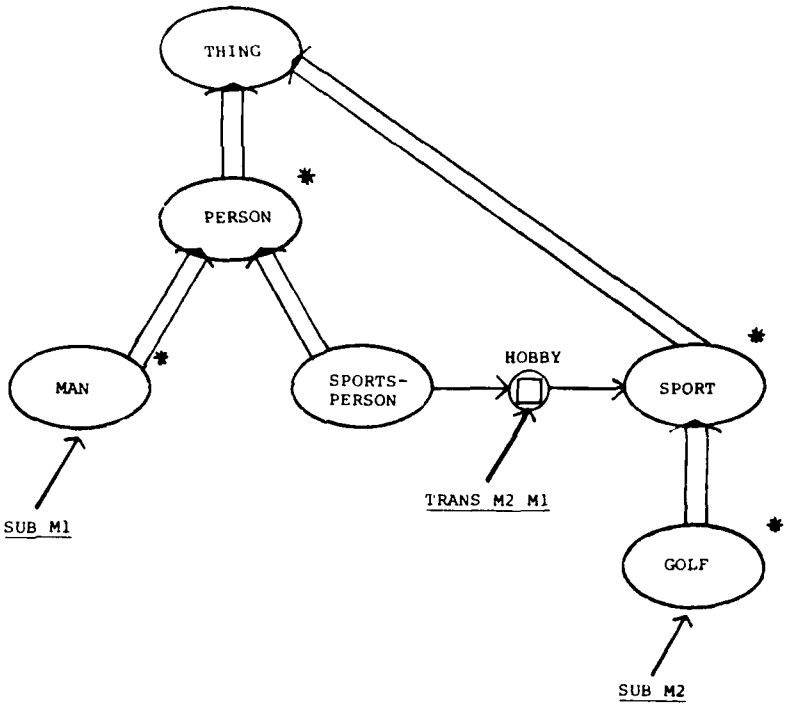


FIG. 18. INITIAL MARKER ASSIGNMENT FOR "A MAN WHOSE HOBBY IS GOLF."

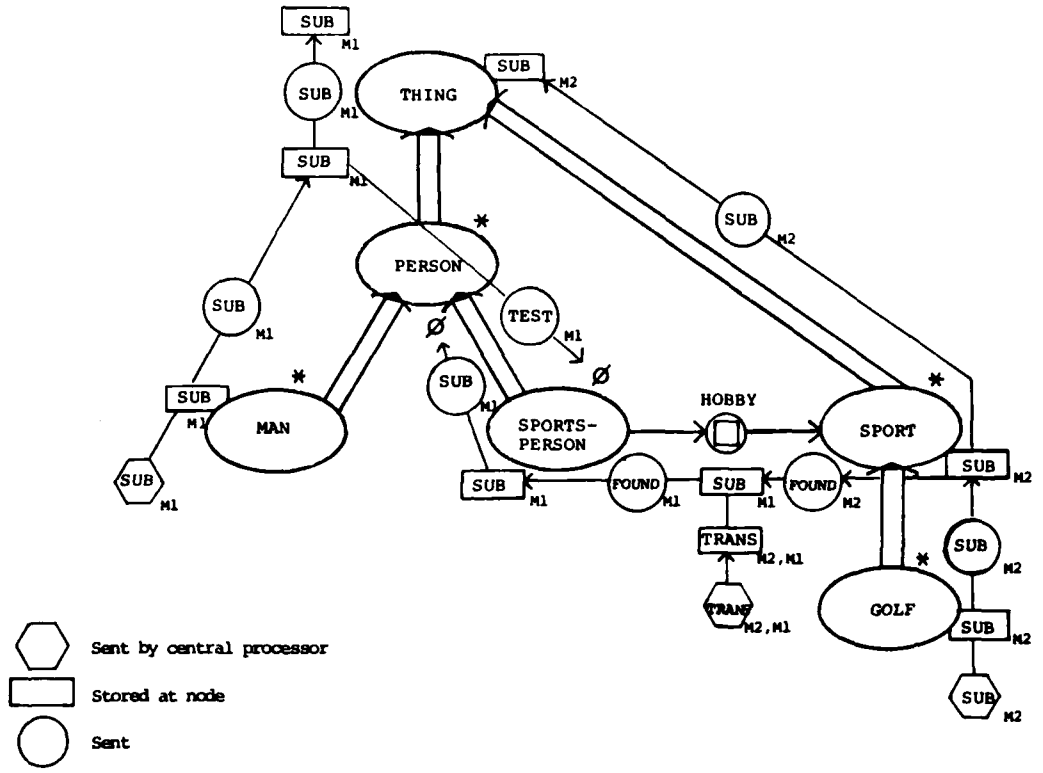


FIG. 19. TRACE OF MARKER PROPAGATION FOR "A MAN WHOSE HOBBY IS GOLF."

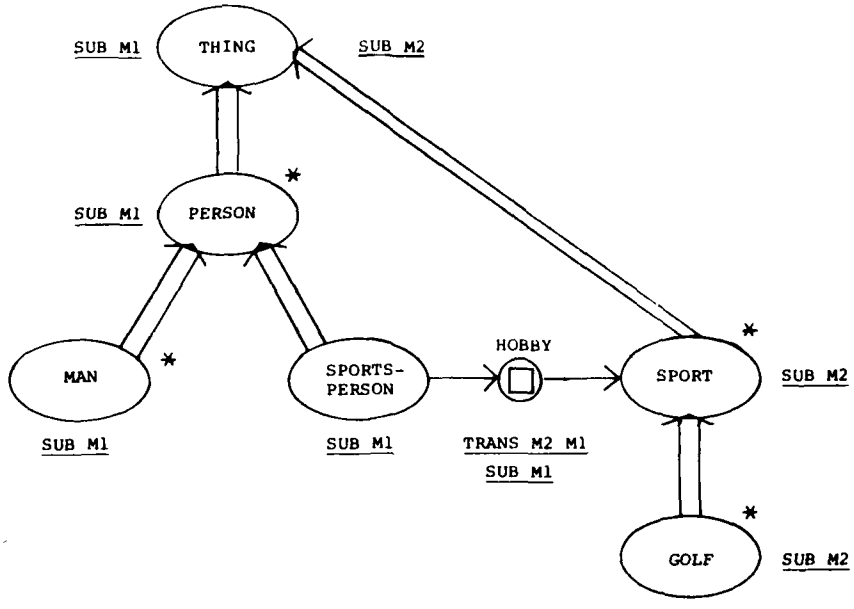


FIG. 20. FINAL CONFIGURATION OF MARKERS FOR "A MAN WHOSE HOBBY IS GOLF."

```
(WHEN A CONCEPT GETS SUB M :
      SEND TO SUPERC NMSS M)

(WHEN A CONCEPT GETS NMSS M:
      STORE NMSS M)

(WHEN A CONCEPT GETS SUB M
      AND DOESN'T HAVE NMSS M AFTER 10 CYCLES :
      STORE MSS M)
 (WHEN A CONCEPT GETS NMSS M
      AND HAS MSS M :
      CANCEL MSS M)
```

These propagation rules have the characteristic that if a SubConcept of a Concept gets marked SUB,m, then that Concept will be marked NMSS,m (for not most specific). If a Concept with SUB,m does not get a message within 10 cycles informing it that one of its SubConcepts is also a subsumer, then it will mark itself with MSS,m - i.e., it will assume that it is a most specific subsumer. In doing so, it will introduce some chance of making an error caused by late discovery that some SubConcept is a subsumer. The chance of error can be reduced by increasing the delay before making the MSS assumption. In any case, the last propagation rule will eventually correct any such false assumptions as the more specific subsumers are found.

6.7 Simulating and Debugging Marker Passing Algorithms

In order to explore the Concept of marker passing described above, we have implemented a simulator that runs on a sequential machine and records the propagation of markers through the network. One can then inspect this record of marker propagation to gain an understanding of the functioning of such algorithms.

Formulating and debugging algorithms of this type is extremely difficult and quite different from conventional methods of programming. Accordingly, it appears necessary to provide the programmer with specialized tools to facilitate this task. In particular, it appears necessary to have a record of the history of marker events that lead to the creation of any given marker. We expect that various graphics displays of marker propagation will help in understanding the operation of marker passing, but we have not yet implemented any such graphics tools.

At the moment, our debugging tools are limited to a display of waves of marker activation by time cycle, a display of causally related chains (and trees) of marker propagation, and a display of the network with its resulting marker configuration. Inspecting these displays to determine what is happening with a marker passing algorithm is only slightly better than looking at core dumps. It is a substantial research task to discover better ways of visualizing what is happening.

6.8 Discussion

There are a number of problems that remain to be worked out in the specification and use of such algorithms. One of them is knowing how long to wait for the answer to develop. As a maximum criterion, one could wait until the network reaches a stable state, at which time all of the appropriate nodes will have been marked. However, in those cases where the results of this

computation are to be passed on to other marker-passing algorithms, it is not necessary to wait for the answer; one can merely set up the propagation conditions so that as each node is marked as an answer, it automatically triggers the propagation of marks for the next computation to use that result. It is difficult to get an intuitive appreciation for the behavior of such a program, but the general characteristic would be that the overall answers to macroscopic questions would be found as soon as possible, starting each subordinate computation as soon as its arguments were available. Final answers could then be reported to the central controller by a request that could be triggered by a node when it received a marker with a particular status.

One difficult problem in the use of such algorithms, which is a variation of the "how long to wait" issue, has to do with the treatment of negation. For example, if we want to set up a condition so that all of the nodes that are marked SUB,ml and do not have a SubConcept that is also marked SUB,ml (i.e., the most specific subsumers) will be marked MSS,ml, we have to decide under what circumstances the negative condition will be assumed to be satisfied. There are at least three ways to deal with such conditions. One is to wait a specified amount of time for the contradicting marker to arrive to cancel the negative condition, and if no such marker arrives in a reasonable amount of time, to consider the negation satisfied. Another is to arrange a separate set of marker propagation conventions that will pass explicitly negative markers indicating that a condition is known

not to be satisfiable, and thus enabling the negative condition. A third is to make possibly false conclusions that can be canceled later when they are found to be false, as in the MSS example above.

There is probably need for all three kinds of negation for various operations in such networks. The first can be handled by the countdown clock provisions at nodes. The second can be done by encoding the negative information using a status code NOT-SUB which will be assigned to a Concept that is proven not to subsume the cspec, and then using the presence of NOT-SUB rather than the absence of SUB as the condition in the rule.

6.9 Conclusions

In this chapter, we have discussed the advantages of parallel implementation of nondeterministic algorithms, we have outlined the basic architecture of an abstract marker passing machine for exploiting such parallelism, we have given an example of a marker passing algorithm for such a machine, and we have discussed some of the issues involved in formulating such algorithms. The approach is not to construct a general purpose parallel engine for arbitrary calculations, but rather to concentrate on a specific class of operations that have wide applicability and high potential for parallelism. This class of problems, referred to as "high level perception" or "situation recognition," lends itself to a high degree of parallelism and is

a central "inner loop" activity in such diverse AI applications as: recognizing a scene, parsing a sentence, recognizing the intent of an utterance, diagnosing a disease, recognizing a standard situation in a planning task, choosing the next rule to apply in a production rule system, recognizing an instance of something of importance (as in an alerting system), and recognizing instances of counterevidence for hypotheses in learning systems.

All of these applications, and many other similar ones that occur throughout applications of artificial intelligence, have the characteristic that potentially many rules, patterns, or templates must be matched against the situation so that the better matching ones are discovered. Moreover, often the problem is more like parsing than matching, in that it involves the synthesis of otherwise unrelated stimuli into a coherent whole. This is the class of operations that I have characterized as "high level perception", and it is to these operations that the parallel architectures being considered are directed.

Because the operations of high level perception are central to a wide range of problems in Artificial Intelligence, because they are the source of many of the computational delays in large systems, because they lend themselves to a high degree of parallel decomposition, and because the marker-passing architectures described here hold the potential for realizing this potential parallelism, the research program outlined in this

paper holds significant promise for realizing real-time symbolic computation for a large class of Artificial Intelligence applications.

7. PUBLICATIONS

To conclude this report, we present here a list of publications by the members of the research group. Also included are presentations given by group members.

Publications

R.J. Bobrow and B.L. Webber, "Some Issues in Parsing and Natural Language Understanding," in Proceedings, 19th Annual ACL Meeting, June 29-July 1, 1981, Stanford University, Stanford, CA.

R.J. Brachman and D.J. Israel, "Some Remarks on the Semantics of Representation Languages" in "Data Modelling in Artificial Intelligence, Databases and Programming Languages" (eds. Brodie, Mylopolous and Schmidt), Springer Verlag, 1981, in preparation.

P.R. Cohen, C.R. Perrault and J. Allen, "Beyond Question Answering" in Strategies for Natural Language Processing, (eds.) W. Lehnart and M. Ringle, Lawrence Erlbaum Assoc., 1981, also BBN report 4644, May 1981.

P.R. Cohen, "The need for identification as a planned action," Proceedings of the International Joint Conference in Artificial Intelligence, 1981.

D.J. Israel and R.J. Brachman, "Distinctions and Confusions: A Catalogue Raisonne" in Proceedings of the International Joint Conference on Artificial Intelligence, pages 452-459 . (IJCAI-81)

C.R. Perrault and P.R. Cohen, "Elements of Discourse Understanding," in It's for your own good: A note on inaccurate reference, (ed.s) Joshi, A., Sag, I., & Webber, B., Cambridge University Press, 1981.

C.L. Sidner and D.J. Israel, "Recognizing Intended Meaning and Speakers' Plans," Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver, B.C. August, 1981, pages 203-208.

W.A. Woods, "Procedural Semantics as a Theory of Meaning," in A.Joshi, I.Sag, and B.Webber (eds.), Elements of Discourse Understanding. Cambridge University Press, 1981, pp. 300-334.

Presentations

M. Bates, "The RUS Parsing System" Special Interest Group on Artificial Intelligence (SIGART) of the ACM, May 1981, Chicago, Ill.

R.J. Bobrow and B.L. Webber, Invited paper for panel: Perspectives on Parsing Issues. 19th Annual ACL Meeting, June 29-July 1, 1981, Stanford University, Stanford, CA.

R.J. Bobrow and B.L. Webber, Parsing and Semantic Interpretation as an Incremental Recognition Process, presented at Symposium on Modeling Human Parsing Strategies, Center for Cognitive Science, University of Texas, Austin, TX, March 1981.

D.J. Israel, "On Self-Modifying Systems." Comments on a panel on Self-Modifying Systems held at IJCAI-81.

D.J. Israel, "Some Issues in Resource Limited Reasoning - The Competence-Performance Distinction in Epistemology", paper read to the Information and Computer Science Department Colloquium at University of Massachusetts, Amherst, November, 1980.

D.J. Israel, "On the Computational Metaphor" (with Brian Smith), talk given at a special session of the Society for the Interdisciplinary Study of the Mind, held in conjunction with the American Philosophical Association, December, 1980.

D.J. Israel, "What Philosophy of Mind Can(not) Learn from AI", paper read to the Philosophy Department Colloquium at Tufts University, January, 1981.

D.J. Israel, "Remarks on Self-Reference and Self-Consciousness", comments on papers by Douglas Hofstadter, Raymond Smullyan and Judson Webb, presented at a special symposium sponsored by the Society for Philosophy and Psychology, April, 1981.

C.L. Sidner, "The Contribution of AI Natural Language Research to Research in Pragmatics of Language," panel presentation at the Seventh International Joint Conference on Artificial Intelligence, Vancouver, B.C., August 25, 1981.

C.L. Sidner, "Recognizing Intended Meaning and Speakers' Plans." presented at the Seventh International Joint Conference on Artificial Intelligence, Vancouver, B.C., August 27, 1981.

C.L. Sidner, "A Model of Speaker Meaning and Speakers' Plans." Presented at Carnegie Mellon University, Computer Science Colloquium, University of Pennsylvania, Computer Science Colloquium, February, 1981, and at University of Massachusetts, Amherst, Computer Science Seminar, April, 1981.

C.L. Sidner, "The Representation of Knowledge in a Computational Approach to the Interpretation of Anaphora," presented at the conference "Models of the Use of Language," University of Bielefeld, West Germany, November 6-8, 1980.

C.L. Sidner, "The Use of Focusing in the Interpretation of Anaphora," Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands, Nov. 10, 1980.

W.A. Woods, "Issues in Knowledge Representation," talk given at Yale University's Cognitive Science Colloquium Series, April 30, 1981.

W.A. Woods, participation on panel for the Naval Research Laboratory on "Applied Computational Linguistics in Perspective," Stanford University, June 1981.

REFERENCES

[1] Allen, J.F.
A plan-based approach to speech act recognition.
Technical Report 131, Department of Computer Science, University of Toronto, January, 1979.

[2] Baker, C.L.
Introduction to Generative-Transformational Syntax.
Prentice-Hall, Inc., Englewood Cliffs NJ, 1978.

[3] Baker, C.L.
Syntactic Theory and the Projection Problem.
Linguistic Inquiry 10(4), 1979.

[4] Bobrow, R.J.
The RUS System.
BBN Report 3878, Bolt Beranek and Newman Inc., 1978.

[5] Bobrow, R.J. and Webber, B.L.
PSI-KLONE - Parsing and Semantic Interpretation in the BBN Natural Language Understanding System.
In CSCSI/CSEIO Annual Conference. CSCSI/CSEIO, 1980.

[6] Bobrow, R.J. and Webber, B.L.
Knowledge Representation for Syntactic/Semantic Processing.
In Proceedings of The First Annual National Conference on Artificial Intelligence. American Association for Artificial Intelligence, 1980.

[7] Bobrow, D.G., and Winograd, T.
An Overview of KRL, a Knowledge Representation Language.
Cognitive Science 1(1), January, 1977.

[8] Brachman, R.J.
A Structural Paradigm for Representing Knowledge.
PhD thesis, Harvard University, May, 1977
Also, BBN Report No.3605, Bolt Beranek and Newman Inc., May 1978.

[9] Brachman, R.J.
Theoretical studies in natural language understanding - Annual Report: 1 May 77 - 30 Apr 78.
BBN Report No. 3888, Bolt Beranek and Newman Inc., September, 1978.

[10] Brachman, R.J.
On the Epistemological Status of Semantic Networks.
In Findler, Nicholas V. (editor), Associative Networks -

The Representation and Use of Knowledge in Computers, . Academic Press, New York, 1979.

[11] Brachman, R.J.
An Introduction to KL-ONE.
In Brachman, R.J., et al. (editors), Research in Natural Language Understanding, Annual Report (1 Sept. 78 - 31 Aug. 79), pages 13-46. Bolt Beranek and Newman Inc, Cambridge, MA, 1979.

[12] Brachman, R.J.
Research in Natural Language Understanding, Quarterly Progress Report No. 7.
BBN Report No. 4190, Bolt Beranek and Newman Inc., May, 1979.

[13] Brachman, R.J., Bobrow, R.J., Cohen, P.R., Klovstad, J.W., Webber, B.L., Woods, W.A.
Research in Natural Language Understanding - Annual Report: 1 Sept 78 - 31 Aug 79.
BBN Report No. 4274, Bolt Beranek and Newman Inc., August, 1979.

[14] Brachman, R.J. and Schmolze, J.
KL-ONE Software Description.
January 1981.

[15] Brachman.
Special Issue on Knowledge Representation.
SIGART Newsletter(70), February, 1980.

[16] Burton, R.R. & Woods, W.A.
A Compiling System for Augmented Transition Networks.
In COLING 76. Sixth International Conference on Computational Linguistics, Ottawa, Canada, June, 1976.

[17] Burton, R. and Brown, J.S.
Semantic Grammar: A Technique for Constructing Natural Language Interfaces to Instructional Systems.
BBN Report 3587, Bolt Beranek And Newman Inc., May, 1977.

[18] Cercone, N.
Representing natural language in extended semantic networks.
Technical Report TR75-11, Department of Computing Science, The University of Alberta, July, 1975.

[19] Charniak, E.
A Common Representation for Problem-Solving and Language Comprehension Information.
Unpublished Manuscript, Computer Science Dept., Brown University, Providence, R.I., 1980.

[20] Cohen, P.R.
On knowing what to say: Planning speech acts.
PhD thesis, University of Toronto, January, 1978.
Technical Report No. 118, Dept. of Computer Science.

[21] Cohen, P.R.
The need for identification as a planned action.
In Proceedings of the International Joint Conference in Artificial Intelligence, pages 31-36. The International Joint Conferences on Artifical Intelligence, The International Joint Conferences on Artifical Intelligence, Vancouver, B.C., August, 1981.

[22] Cohen, P.R., & Perrault, C.R.
Elements of a plan-based theory of speech acts.
Cognitive Science 3:177-212, 1979.

[23] Cohen, P.R., Perrault, C.R., Allen, J.
Beyond Question Answering.
In W. Lehnart and M. Ringle (editors), Strategies for Natural Language Processing, . Lawrence Erlbaum Assoc., Hillsdale, NJ, 1981.

[24] Fahlman, S.E.
NETL: A System for Representing and Using Real-World Knowledge.
MIT Press, Cambridge, Massachusetts, 1979.

[25] Fodor, J.A., Garrett, M.F., Walker, E.C.T., and Parkes, C.H..
Against Definitions.
Cognition 8(3):263-367, 1980.

[26] Genesereth, M.R.
Automated consultation for complex computer systems.
PhD thesis, Department of Computer Science, Division of Applied Sciences, Harvard University, September, 1978.

[27] Genesereth, M.R., Greiner, R., and Smith, D.
MRS Manual
Department of Computer Science, Stanford University, 1980.
Memo HPP-80-24.

[28] Greenfeld, N.R. and Yonke, M.D.
AIPS: an Information Presentation System for Decision Makers.
Report No. 4228, Bolt Beranek and Newman Inc., Dec. 1979.

[29] Greiner, R. and Lenat, D.
A Representation Language Language.
In Proceedings of The First Annual National conference on Artificial Intelligence, pages 165-169. Stanford University, August, 1980.

Sponsored by The American Association for Artificial Intelligence.

[30] Grice, H.P.
Meaning.
Philosophical Review 66:377-388, 1957.

[31] Grice, H.P.
Utterer's Meaning and Intentions.
Philosophical Review 68(2):147-177, 1969.

[32] Halliday, M. & Hasan, R.
Cohesion in English.
Longman's, 1977.

[33] Hayes, P.J.
The Logic of Frames.
In Metzing, Dieter (editor), Frame Conceptions and Text Understanding, pages 46-61. Walter de Gruyter and Co., Berlin, 1979.

[34] Israel, D.J., and Brachman, R.J.
Distinctions and Confusions: A Catalogue Raisonne.
In Proc. IJCAI-81, pages 452-459. International Joint Conference on Artificial Intelligence, Vancouver, British Columbia, 1981.

[35] Mackworth, A.K.
Consistency in Networks of Relations.
Artificial Intelligence 8:99-118, 1977.

[36] M. Marcus.
A Theory of Syntactic Recognition for Natural Language.
MIT Press, CambridgeMA, 1980.

[37] Mark, W. S. and Barton, G. E.
The RUSGrammar Parsing System.
GMR 3243, General Motors Research Laboratories, 1980.

[38] Minsky, M.
A Framework for Representing Knowledge.
In Winston, P. H. (editor), The Psychology of Computer Vision, pages 211-277. McGraw-Hill Book Company, New York, 1975.

[39] T. Mitchell.
An Analysis of Generalization as a Search Problem.
In Proceedings of 6-IJCAI, pages 577-582. International Joint Conference on Artificial Intelligence, 1979.

[40] Newell, A.
The Knowledge Level.

The Journal of the American Association for Artificial Intelligence 1(3):??-??, ?, 1981.

[41] Norman, D. and Bobrow, D.
On Data-limited and Resource-limited Processes.
Technical Report CSL74-2, Xerox PARC, May, 1974.

[42] Ochs, E., Schieffelin, B.B., and Platt, M.L.
Propositions across utterances and speakers.
In Ochs, E., & Schieffelin, B. B. (editors), Developmental Pragmatics, . Academic Press, New York, 1979.

[43] Ochsman, R.B, and Chapanis, A.
The effects of 10 communication modes on the behaviour of teams during co-operative problem-solving.
International Journal of Man-Machine Studies 6(5):579-620, Sept., 1974.

[44] W.H. Paxton.
A Framework for Speech Understanding.
Technical Report 142, SRI International, June, 1977.

[45] Perrault, C.R., and Allen, J.F.
A plan-based analysis of indirect speech acts.
American Journal of Computational Linguistics 6(3):167-182, 1980.

[46] Perrault, C.R. and Cohen, P.R.
It's for your own good: A note on inaccurate reference.
In Joshi, A., Sag, I., & Webber, B. (editors), Elements of Discourse Understanding, . Cambridge University Press, Cambridge, Mass., 1981.

[47] Putnam, H.
Is Semantics Possible?
In Schwartz, S. P. (editor), Naming, Necessity, and Natural Kinds, pages 102-118. Cornell University Press, Ithaca, 1977.

[48] Quillian, M.R.
Semantic Memory.
Report No. AFCRL-66-189, Bolt Beranek and Newman Inc., 1966.

[49] Quillian, M.R.
Semantic Memory.
In M. Minsky (editor), Semantic Information Processing, pages 27-70. M.I.T. Press, Cambridge, MA, 1968.

[50] Quillian, M.R.
The Teachable Language Comprehender.
CACM 12:459-475, 1969.

[51] Quillian, M.R. and Collins, A.M.
Retrieval Time from Semantic Memory.
Journal of Verbal Learning and Verbal Behavior 8, 1969.

[52] Reiter, R.
On Reasoning by Default.
In Proceedings of TINLAP-2, pages 210-218. University of Illinois at Urbana-Champaign, July, 1978.

[53] Roberts, R.B., Goldstein, I.P.
The FRL Manual
Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, 1977.
MIT AI Lab Memo 409.

[54] Robinson, J.
DIAGRAM: A grammar for dialogues.
Technical Report 205, SRI International, 1980.

[55] Sacerdoti, E.
A Structure for Plans and Behavior.
American Elsevier, New York, 1977.

[56] Schmidt, D.F., Sridharan, N.S., and Goodson, J.L.
The plan recognition problem: An intersection of artificial intelligence and psychology.
Artificial Intelligence 10:45-83, 1979.

[57] Schubert, L.K.
Extending the expressive power of semantic networks.
Artificial Intelligence 7(2):163-198, Summer, 1976.

[58] Schubert, L.K., Goebel, R.G., and Cercone, N.J.
The Structure and Organization of a Semantic Net for Comprehension and Inference.
In Findler, N.V. (editor), Associative Networks: Representation and Use of Knowledge by Computers, pages 121-175. Academic Press, New York, 1979.

[59] Searle, J.R.
Speech acts: An essay in the philosophy of language.
Cambridge University Press, Cambridge, 1969.

[60] Shapiro, S.C.
An introduction to SNePS.
Technical Report, Computer Science Department, Indiana University, March, 1975.

[61] Sidner, C.L.
Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse.
Technical Report 537, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June, 1979.

PhD. Thesis.

[62] Sidner, C.L.
Protocols of users manipulating visually presented information with natural language.
May 1980.
Unpublished manuscript, Bolt Beranek and Newman, Inc.

[63] Smith, B.
First Glimpse of MANTIQ.
1980.

[64] M. Stefik.
Planning with Constraints.
Technical Report STAN-CS-80-784, Stanford Computer Science Department, 1980.

[65] Sussman, G.
A Computer Model of Skill Acquisition.
American Elsevier, New York, 1975.

[66] Van Lehn, K.
Determining the Scope of English Quantifiers.
Technical Report 483, MIT Artificial Intelligence Laboratory, 1978.

[67] Waltz, D.L.
Understanding Line Drawings of Scenes with Shadows.
In Winston, P. (editor), The Psychology of Computer Vision, pages 19-91. McGraw Hill, New York, NY., 1975.

[68] Webber, B.L.
A formal approach to discourse anaphora.
BBN Report 3761, Bolt Beranek and Newman, May, 1978.

[69] Webber, B.L.
So What Can We Talk About Now.
In M. Brady (editor), Computational Approaches to Discourse, . MIT Press, 1982.

[70] Wilensky, R.
Understanding goal-based stories.
PhD thesis, Department of Computer Science, Yale University, 1978.
Research Report No. 140.

[71] Woods, W.A.
Transition Network Grammars for Natural Language Analysis.
CACM 13(10), October, 1970.

[72] Woods, W.A., Kaplan, R.M. and Nash-Webber, B.L.
The Lunar Sciences Natural Language Information System: Final Report.

BBN Report 2378, Bolt Beranek and Newman Inc., June, 1972.

[73] Woods, W.A.
Semantics and Quantification in Natural Language Question Answering.
In M. Yovits (editor), Advances in Computers, pages 1-87. Academic Press, 1978.

[74] Woods, W.A.
Taxonomic Lattice Structures for Situation Recognition.
In Theoretical Issues in Natural Language Processing - 2. ACL and ACM/SIGART, July, 1978.

[75] Woods, W.A.
Research in natural language understanding, Quarterly Technical Progress Report No.6: Parallel Algorithms for Real Time Knowledge Based Systems.
BBN Report No. 4181, Bolt Beranek and Newman Inc., February, 1979.

[76] Woods, W.A.
Cascaded ATN Grammars.
Amer. J. Computational Linguistics 6(1):1-15, Jan.-Mar., 1980.

[77] Woods, W.A.
A Proposal for Research in Natural Language Understanding.
Proposal No. P77-ISD-6A, Bolt Beranek and Newman Inc, 1977.

[78] Woods, W.A.
Research in Natural Language Understanding, Quarterly Progress Report No. 4: Generalizations of ATN Grammars.
Report No. 3963, Bolt Beranek and Newman Inc., August, 1978.

Official Distribution List

Contract N00014-77-C-0378

	Copies
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Office of Naval Research Information Systems Program Code 437 Arlington, VA 22217	2
Office of Naval Research Code 200 Arlington, VA 22217	1
Office of Naval Research Code 455 Arlington, VA 22217	1
Office of Naval Research Code 458 Arlington, VA 22217	1
Office ot Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	1
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, IL 60605	1
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, CA 91106	1
Office of Naval Research New York Area Office 715 Broadway - 5th Floor New York, NY 10003	1

Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20380	6
Naval Ocean Systems Center Advanced Software Technology Division Code 5200 San Diego, CA 92152	1
Dr. A.L. Slafkosky Scientific Advisor Commandant of the Marine Corps (Code RD-1) Washington, D.C. 20380	1
Mr. E.H. Gleissner Naval Ship Research & Dev. Center. Computation & Mathematics Dept. Bethesda, MD 20084	1
Capt. Grace M. Hopper NAICOM/MIS Planning Branch (OP-916D) Office of Chief of Naval Operations Washington, D.C. 20350	1
Mr. Kin B. Thompson NAVDAC 33 Washington Navy Yard Washington, D.C. 20374	1
Advanced Research Projects Agency Information Processing Techniques 1400 Wilson Boulevard Arlington, VA 22209	1
Capt. Richard L. Martin, USN Commanding Officer USS Francis Marion (LPA-249) FPO New York 09501	1
Director National Security Agency Attn: R54, Mr. Page Fort G.G. Meade, MD 20755	1
Director National Security Agency Attn: R54, Mr. Glick Fort G.G. Meade, MD 20755	1

Major James R. Kreer 1
Chief, Information Sciences
Dept. of the Air Force
Air Force Office of Scientific
Research
European Office of Aerospace
Research and Development
Box 14
FPO New York 09510

Dr. Martin Epstein 1
National Library of Medicine
Bldg. 38A, 8th Floor Lab
8600 Rockville Pike
Bethesda, MD 20209